In compliance with the
Canadian Privacy Legislation
some supporting forms
may have been removed from
this dissertation.

While these forms may be included
in the document page count,
their removal does not represent
any loss of content from the dissertation.

University of Alberta

Predicting Protein Sub-cellular Localization from Homologs Using Machine Learning Algorithms

by

Zhiyong Lu

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Fall 2003

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisisitons et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-612-87817-1
Our file Notre référence
ISBN: 0-612-87817-1

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou aturement reproduits sans son autorisation.

Canada

University of Alberta

Library Release Form

Name of Author: Zhiyong Lu

Title of Thesis: Predicting Protein Sub-cellular Localization from Homologs Using Machine Learning Algorithms

Degree: Master of Science

Year this Degree Granted: 2003

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as herein before provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

Date: August 18, 2003

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Predicting Protein Sub-cellular Localization from Homologs Using Machine Learning Algorithms** submitted by Zhiyong Lu in partial fulfillment of the requirements for the degree of **Master of Science**.

Dr. Duane Szafron
Co-Supervisor

Dr. Russ Greiner
Co-Supervisor

Dr. Paul Lu

Dr. Warren Gallin

Date: August 11, 2003

Abstract

With the rapid growth of modern sequencing technology, more and more entire genomes have been completed, leading to an explosion of new gene sequences. Unfortunately, for most of these new sequences, we do not know any of their properties, such as their sub-cellular localizations. It is too time consuming to determine the properties of each sequence manually in a biological laboratory, since it typically takes months or even years to determine the properties of even a single protein sequence.

A much quicker alternative is to use computational techniques to make predictions in a high-throughput fashion. Proteome Analyst SUB-cellular localization server (PA-SUB) is a system that uses machine learning techniques to predict the sub-cellular localization of each protein in a proteome. This dissertation demonstrates how PA-SUB applies established machine learning techniques to make accurate predicitons. It describes techniques and experiments that establish reliable training/testing datasets, significant feature extraction, efficient algorithm selection and convincing validation methods.

PA-SUB is described and the results of its application are presented along with concrete examples. Experiments are presented that compare different feature identification techniques and different classifier technologies. We obtained execellent results (approximately 90% 5-fold cross-validation accuracy) for sub-cellular localization prediction. These results are better than published results using other techniques. More generallly, this dissertation demonstrates that computational machine learning techniques can be used effectively for the prediction of protein properties.

Acknowledgements

I'd love to thank:

- My parents for their support towards my graduate study.
- My supervisors, Duane Szafron and Russ Greiner, for leading me to this research and many helpful comments.
- Paul Lu for his valuable discussion
- Warren Gallin for being my external examiner and helpful feedback
- All other students in the PA group for the great time at the bioinformatics lab

Contents

1 Introduction 1
- 1.1 Introduction to Protein Sub-cellular Localization . . . 1
- 1.2 Related Work . . . 5
  - 1.2.1 Prediction Based on Amino Acid Composition . . . 5
  - 1.2.2 Prediction Based on N-terminal Signals . . . 6
  - 1.2.3 Prediction Based on Similarity Search . . . 8
  - 1.2.4 Combination System . . . 9
  - 1.2.5 Limitations of Existing Systems . . . 11
- 1.3 Research Goal . . . 13
- 1.4 Summary . . . 13

2 The Prediction Process 14
- 2.1 A 1-Nearest-Neighbor Prediction . . . 14
- 2.2 Building a Classifier . . . 15
- 2.3 Using a Classifier for Prediction . . . 17
- 2.4 Classifier Evaluation . . . 18
- 2.5 Summary . . . 20

3 Machine Learning and Classification 21
- 3.1 Classification: A Two-Step Process . . . 21
- 3.2 k Nearest Neighbors . . . 22
- 3.3 Naïve Bayes . . . 23
- 3.4 Tree-augmented Naïve Bayes . . . 25
- 3.5 Artificial Neural Nets . . . 27
- 3.6 Support Vector Machines . . . 30
- 3.7 Summary . . . 31

4 Feature Extraction and Selection 33
- 4.1 Feature Extraction . . . 33
  - 4.1.1 Introduction . . . 33
  - 4.1.2 The Number of PSI-BLAST Iterations . . . 34
  - 4.1.3 The Number of Homologs to Use . . . 35
  - 4.1.4 Selecting Swiss-Prot Fields . . . 36
  - 4.1.5 Words or Phrases . . . 37
  - 4.1.6 Experimental Results . . . 38
- 4.2 Feature Selection - Wrappers . . . 39
  - 4.2.1 Introduction . . . 39

4.2.2 The Information Content 39
4.2.3 The Wrapper Model 40
4.3 Summary 41

5 Experiments **42**
5.1 The Dataset 42
5.2 k Nearest Neighbors 44
5.3 PA-SUB Accuracy 45
5.4 *1-Organism* Classification 48
5.5 Comparisons with Other Methods 51
5.5.1 Comparion with LOCkey 51
5.5.2 Comparison with PSORT-B 53
5.6 Summary 53

6 Discussion and Future Work **55**
6.1 PA-SUB Coverage 55
6.2 Future Directions 56
6.2.1 Pattern Discovery 56
6.2.2 Gene Ontology Classification Scheme 57
6.2.3 Parallelism 57
6.3 Contributions 57

Bibliography **58**

List of Figures

1.1 A primary protein sequence from the Swiss-Prot database 2
1.2 The cell structures of Gram-positive bacteria and Gram-negative bacteria. 3
1.3 The cell structure of eukaryotics includes many different organelles: peroxisome, chloroplast, thylakoid, mitochondrion, plasma membrane, secreted, golgi apparatus, endoplasmatic reticulum, nucleus, cytoplasm, lysosome and endosome. 3
1.4 The overall differences in the amino acid composition between all groups appear strong enough for prediction purposes (NNPSL 1998). 6
1.5 LOCkey algorithm. 9

2.1 The training and predicting phrases of classification in PA-SUB . . . 15
2.2 PA-SUB builds a classifier using machine learning (ML) algorithm. . 16
2.3 PA-SUB makes a prediction using pre-built classifier. 17
2.4 PA-SUB predicts location "extracellular" with probability .932 for the example sequence (Figure 1.1) using a pre-built animal classifier. 18

3.1 A Two-Step process: learning and evaluation. 21
3.2 An example of a simple Naïve Bayes classifier. 24
3.3 An example of Naïve Bayes classifier structure. Here, nodes are variables and links between nodes represent the causal dependency. . . . 25
3.4 An example of Tree-augmented Naïve Bayes classifier. Under this model, we allow some additional edges between attributes for representing simple correlations between the attributes. 26
3.5 An example of a Multi-Layer Neural Network. It has three layers including input layer, hidden layer and output layer. Each • unit above represents a neuron. 28
3.6 A unit. 28
3.7 A linear classifier is defined by a hyperplane's normal vector **w** and an offset b, i.e. the decision boundary is $\{x|(w \cdot x) + b = 0\}$. Each of the two halfspaces defined by this hyperplane corresponds to one class, i.e. $f(x) = sign((w \cdot x) + b)$. The objects that are circled are called *support vectors* [21]. 30
3.8 A two dimensional classification example. Using the second order monomials $x_1^2, \sqrt{2}x_1x_2, x_2^2$ as features a separation in feature space can be found using a *linear* hyperplane (right). The input space is a non-linear dataset (left) [21]. 31

4.1 Computing features for a protein sequence in PA-SUB 34
4.2 An example of sequence alignment. For simplicity, we allow only 4 letters to appear: A(alanine), C(cysteine), G(glycine), and T(threonine). Coincidentally, these also happen to be the letters used for four nucleotides(adenine, cytosine, guanine, thymine) of DNA. 35
4.3 Extracting tokens from a Swiss-Prot entry in PA-SUB 37

List of Tables

1.1 The twenty amino acids found in living proteins 2

1.2 The ontologies across five different taxonomic categories including animal, plant, fungi, Gram-positive bacteria(GP) and Gram-negative bacteria(GN). Abbreviations for localizations: nuc(lear), end(oplasmic reticulum), gol(gi), mit(ochondria), pe(ro)x(isomal), lys(osomal), cyt(oplasmic), mem(brane), ext(racellular), chl(oroplast), vac(uole), inn(er membrane), per(iplasmic), (cell) wal(l) and out(er membrane). 4

1.3 Accuracies and representative coverage of current sub-cellular localization predictors 12

2.1 Confusion matrix for the Gram-negative bacteria classifier, trained on Swiss-Prot data. The abbreviations of ontological labels are shown in Table 1.2. Column no_pred(iction) indicates the number of the sequences that do not have any homologs in the PSI-BLAST search. 19

3.1 Confusion matrix for the 1NN classifier, trained on GN bacteria data from Swiss-Prot. 23

3.2 Confusion matrix for the TAN classifier, trained on GN bacteria data from Swiss-Prot. 27

3.3 Confusion matrix for the ANN classifier, trained on GN bacteria data from Swiss-Prot. 30

3.4 Confusion matrix for the SVM classifier, trained on GN bacteria data from Swiss-Prot. 32

3.5 A comparison of the statistics of all four machine learning algorithms, built using the same dataset, GN bacteria from Swiss-Prot. The ontological labels are: (cyt)oplasmic, (ext)racellular, (per)iplasmic, (inn)er membrane, cell (wal)l and (out)er membrane. 32

4.1 A comparison of the accuracy of some Gram-negative classifiers, which use different homolog selection techniques (number of PSI_BLAST iterations and number of top homologs), different Swiss-Prot fields (KW, IPR and SCELL) and different feature extraction techniques (semi-colon delimited phrases or words for KW, and fixed phrases or words for SCELL). 38

4.2 The information content of eight features ($F_1 ... F_8$) in eight sequences. Each class has two protein instances. Feature values are either '1' or '0' in a sequence. Information content of each feature shown in the last row is computed using Formula 4.1. 40

4.3 A comparison of the 5-fold cross-validation accuracies before and after using feature selection technique, the wrapper model, with different machine learning algorithms. The percentage of used features is given. It used the same dataset shown in Table 2.1. 41

5.1 The number of training sequences are extracted by following the rules across five different categories. 44
5.2 A comparison of the accuracy of three nearest neighbor classifiers (1NN, 3NN and 5NN) and the Naïve Bayes (NB) classifier on the five categories of Swiss-Prot data, the LOCkey [22] and the PSORT-B [13] data. 44
5.3 Confusion matrix for the PA-SUB **animal** classifier, trained on Swiss-Prot data. 45
5.4 Confusion matrix for the PA-SUB **plant** classifier, trained on Swiss-Prot data. 46
5.5 Confusion matrix for the PA-SUB **fungi** classifier, trained on Swiss-Prot data. 46
5.6 Confusion matrix for the PA-SUB **Gram-positive bacteria** classifier, trained on Swiss-Prot data. 47
5.7 Confusion matrix for the PA-SUB **Gram-negative bacteria** classifier, trained on Swiss-Prot data. 47
5.8 Confusion matrix for the 1-organism classifier, trained on animal dataset from Swiss-Prot. The 1-organism is **Bos taurus (Bovine)**. 49
5.9 Confusion matrix for the 1-organism classifier, trained on plant dataset from Swiss-Prot. The 1-organism is **Zea mays (Maize)**. 49
5.10 Confusion matrix for the 1-organism classifier, trained on fungi dataset from Swiss-Prot. The 1-organism is **Neurospora crassa**. 50
5.11 Confusion matrix for the 1-organism classifier, trained on Gram-positive bacteria dataset Swiss-Prot. The 1-organism is **Haemophilus influenzae**. 50
5.12 Confusion matrix for the 1-organism classifier, trained on Gram-negative bacteria from Swiss-Prot. The 1-organism is **Streptomyces coelicolor**. 50
5.13 A comparison of the statistics of a PA-SUB custom classifier built using the LOCkey 1161 sequence training data with the statistics produced by their classifier on their training data. 52
5.14 Confusion matrix for a custom NB classifier, trained on LOCkey 3146 protein sequences (100% coverage of test data). 52
5.15 A comparison of the statistics of a PA-SUB custom classifier built using the PSORT-B training data with the statistics produced by the PSORT-B predictor, built from the same training data. The ontological labels are: (cyt)oplasmic, (inn)er membrane, (per)iplasmic, (out)er membrane and (ext)racellular. 53

6.1 The current PA Coverage across five different categories. The number of instances and no_prediction are from Table 5.3 to Table 5.7 of Chapter 5. 55

6.2 Coverage of the PA-SUB classifier on some fully sequenced organisms. The count is the total number of genetic sequences for the organism. An excluded sequence (excluded) is one for which PA-SUB was unable to find at least one homolog whose E-value was less than .001 that contained at least one relevant feature. Coverage is the percentage of all non-excluded sequences from an organism. 56

Chapter 1

Introduction

This dissertation describes a solution to the problem of predicting protein sub-cellular localization in a high-throughput fashion using established machine learning techniques.

Chapter 1 describes the problem of protein sub-cellular localization prediction and presents the current state of the art. Chapter 2 introduces the classification process for genetic sequences. It describes how a classifier is built, how it makes a prediction and how it can be evaluated. Chapter 3 introduces some of the established machine learning algorithms we have evaluated. Chapter 4 discusses our strategies of extracting and selecting features for the machine learning algorithms. Chapter 5 describes our experiments and presents the results. Chapter 6 provides some concluding remarks and gives some discussion on future research directions.

1.1 Introduction to Protein Sub-cellular Localization

A protein is a macromolecule composed of chains of smaller molecules called **amino acids**. There are twenty different primary amino acids (Table 1.1) found in the proteins of living cells. Proteins are in cells, which are in our hair, nails, muscle and many other places inside the body. They have many important functions for human life such as support, structure, movement, transport, communication, and disease defense. For example, enzymes are perhaps the most interesting of all proteins. They work actively in cells as catalysts, speeding up chemical activities. In this dissertation, protein sequences are represented as sequences of letters, where each letter represents a single amino acid. Such a sequence is referred to as a **primary protein sequence**. The sequence length ranges from 17 to 4936 amino acid residues long, with average length about 322 $\pm$ 280 letters long. Figure 1.1 shows the start

Table 1.1: The twenty amino acids found in living proteins

	Name	Three-letter code	One-letter code
1	Alanine	Ala	A
2	Cysteine	Cys	C
3	Aspartic Acid	Asp	D
4	Glutamic Acid	Glu	E
5	Phenylalanine	Phe	F
6	Glycine	Gly	G
7	Histidine	His	H
8	Isoleucine	Ile	I
9	Lysine	Lys	K
10	Leucine	Leu	L
11	Methionine	Met	M
12	Asparagine	Asn	N
13	Proline	Pro	P
14	Glutamine	Gln	Q
15	Arginine	Arg	R
16	Serine	Ser	S
17	Theronine	Thr	T
18	Valine	Val	V
19	Tryptophan	Trp	W
20	Tyrosine	Tyr	Y

of one animal protein from a rabit that was obtained from the Swiss-Prot protein database [4].

```
> VTNC_RABIT
MAPLRPIFTLALLLWVVLADQESCKDRCTEGFNANRKCQCDELCSYYQSCCAD
YAAECKPQVTRGDVFTMPEDEYGPYDYIEQTKDNASVHAQPESPTVGQEPTLS
...
```

Figure 1.1: A primary protein sequence from the Swiss-Prot database

Protein sub-cellular localization is a key functional characteristic of proteins. In order to function properly, proteins needs to be at various localization sites within the cell. Knowledge of sub-cellular localization helps scientists to select proteins for more thorough investigation. One example is drug discovery. Sub-cellular localization also influences the design of experimental strategies for functional characterization. For example, it might be helpful to check ATP binding for a cytosolic protein. However, in the case of extracellular proteins, proteolytic digestion assays appear much more appropriate. Since it is so important to know the protein

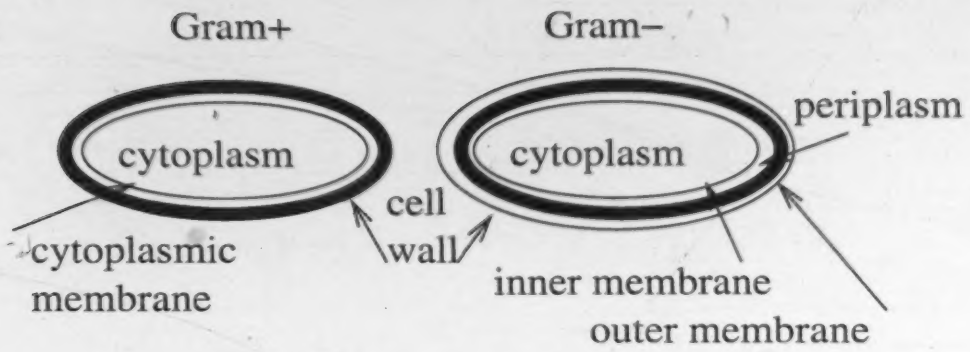

Figure 1.2: The cell structures of Gram-positive bacteria and Gram-negative bacteria.

sub-cellular localization, biologists have been conducting laboratory experiments for some time now. The challenge today is that with the rapid growth of modern sequencing technology, resulting in more than 1200 genome sequences deposited in public databases, it is too time consuming to determine the localization property of each protein sequence manually. Therefore, automated annotation tools are needed.

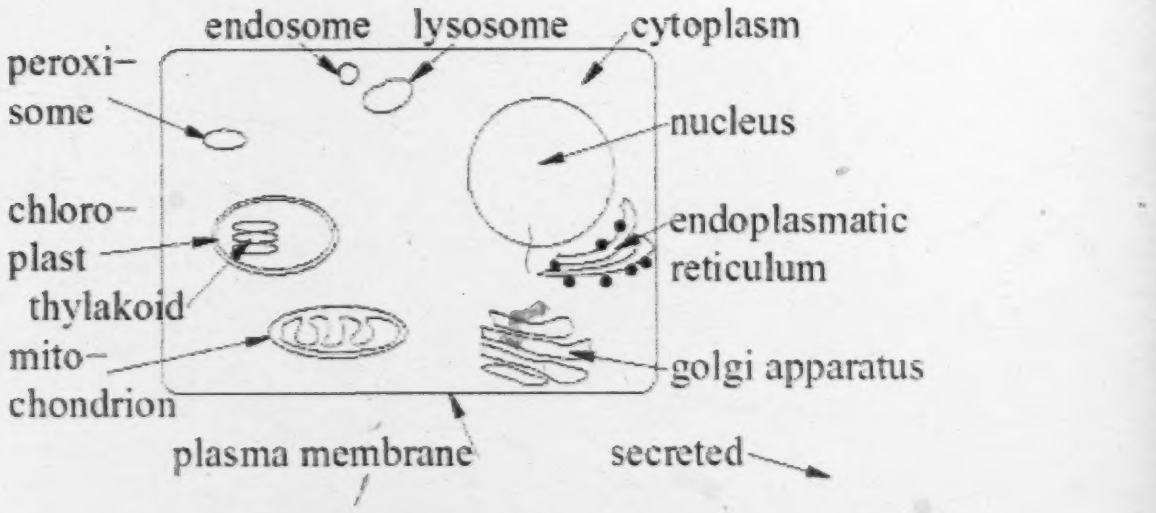

Figure 1.3: The cell structure of eukaryotics includes many different organelles: peroxisome, chloroplast, thylakoid, mitochondrion, plasma membrane, secreted, golgi apparatus, endoplasmatic reticulum, nucleus, cytoplasm, lysosome and endosome.

Proteins may be localized at various locations within the cell or transported to

the extracellular space. The set of locations depend on the type of a cell. Living cells are divided into two types - prokaryotic and eukaryotic. This division is based on the internal complexity of the cell. Protein sorting, i.e. the processes through which proteins are routed to their proper final destination within a cell, is simplest in Gram-positive prokaryotes, where proteins are located only in the cytoplasm, the membrane, or the cell wall or are secreted from the cell as shown in Figure 1.2. Protein locations in Gram-negative bacteria include the cytoplasm, the inner membrane, the periplasm, the outer membrane, the cell wall and the extracellular space. There are many more important locations in eukaryotic proteins, due to the presence of membrane-bound organelles. The major location sites for eukaryotes are shown in Figure 1.3: peroxisome, chloroplast, thylakoid, mitochondrion, plasma membrane, secreted, golgi apparatus, endoplasmatic reticulum, nucleus, cytoplasm, lysosome and endosome. Others that are not shown include vacuoles and cell wall.

Table 1.2: The ontologies across five different taxonomic categories including animal, plant, fungi, Gram-positive bacteria(GP) and Gram-negative bacteria(GN). Abbreviations for localizations: nuc(lear), end(oplasmic reticulum), gol(gi), mit(ochondria), pe(ro)x(isomal), lys(osomal), cyt(oplasmic), mem(brane), ext(racellular), chl(oroplast), vac(uole), inn(er membrane), per(iplasmic), (cell) wal(l) and out(er membrane).

Category	Sub-cellular Localizations									
Animal	nuc,	end,	gol,	mit,	pex,	lys,	cyt,	mem,	ext	
Plant	nuc,	end,	gol,	mit,	pex,	chl,	vac,	cyt,	mem,	ext
Fungi	nuc,	end,	gol,	mit,	pex,	vac,	cyt,	mem,	ext	
GP	cyt,	wal,	mem,	ext						
GN	cyt,	inn,	per,	wal,	out,	ext				

Since classification requires a dictionary of class labels, a controlled vocabulary or ontology is required for sub-cellular localization. PA-SUB supports five different taxonomic categories: animal, plant, fungi, Gram-negative bacteria (GN) and Gram-positive bacteria (GP), which are based on the PSORT ontologies [17].

Table 1.2 lists all the ontological class labels for each of the five categories. As shown, the ontology for animal, fungi and plant are quite close, except that plants have chloroplasts and vacuoles and lack lysosomes compared to animals; fungi lacks the chloroplasts of plants. GN differs from GP by having two membranes: inner and outer membrane, and a periplasm.

1.2 Related Work

A number of systems have been developed over the past few years that support automated prediction of sub-cellular localization using a series of prediction algorithms, based on amino acid sequence information. There are three basic approaches.

1.2.1 Prediction Based on Amino Acid Composition

It has been known that the amino acid composition of a protein can be an indicator of its sub-cellular localization. Previous studies [23] have shown that intra-cellular and extracellular proteins differ significantly in their amino acid composition and these differences are strong enough to be used as the basis for a prediction method. NNPSL [3] was designed to infer protein sub-cellular localizations based on amino acid composition. It computed the average fraction for each amino acid of Table 1.1 and its standard error for all sub-cellular localization. The results are presented in Figure 1.4. These results show that only F, H, M and W have minor fluctuations, while the other amino acids show strong differences in composition between different sub-cellular localizations. For example, amino acids R and S differ substantially between the total seven locations. To determine how effective a measure the amino acid composition is, NNPSL applied neural networks [20] to distinguish different sub-cellular localizations. It could predict three possible sub-cellular localizations in prokaryotics (Periplasmic, Cytoplasmic and Extracellular) and four localizations in eukaryotics (Extracellular, Cytoplasmic, Mitochondrial and Nuclear).

Besides NNPSL, ProtLock [5] and SubLoc [16] are two other automatic methods for assignment of the sub-cellular localization based on amino acid composition information. ProtLoc uses statistical analysis and handles eukaryotic and prokaryotic sequences together. SubLoc follows the lead of NNPSL's ontologies on the sub-cellular localizations (three locations in prokaryotics and four in eukaryotics) and applies a support vector machine (SVM) algorithm to abtain better prediction accuracy.

NNPSL has a web-based prediction server, which always yields the top two predicted localization sites. We submitted the eukaryotic sequence in Figure 1.1 (whose sub-cellular localization is 'extracellular') to NNPSL and obtained predicted results: 'extracellular' as its first predicted site and 'nuclear' as its second. It also shows us the 'Expected Prediction Accuracy (EPA)' for its first choice and its combined

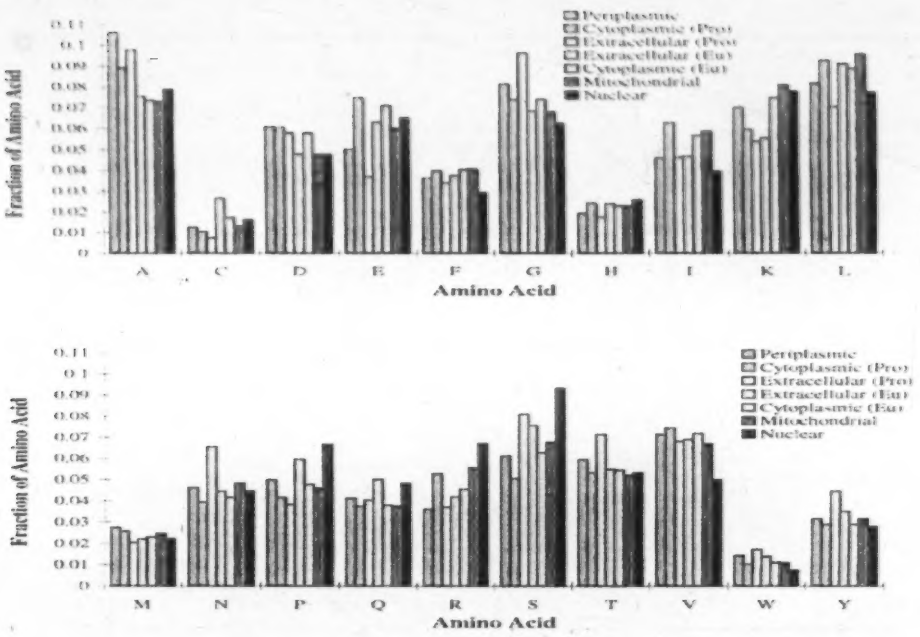

Figure 1.4: The overall differences in the amino acid composition between all groups appear strong enough for prediction purposes (NNPSL 1998).

choices. This is the output from NNPSL:

Sequence Name	Location(1st)	Location(2nd)	EPA(1st)	EPA(1st or 2nd)
test	extracellular	nuclear	51.1%	91.4%

NNPSL is 51.1% sure about its first location site (extracellular) and 91.4% confident about the sequence being either extracellular or nuclear.

1.2.2 Prediction Based on N-terminal Signals

Sub-cellular protein sorting is a fundamental aspect of cellular life [24]. In many cases, sorting depends on 'signals', which are short sub-sequences of approximately 3 to 70 amino acids, and can be identified by looking at the primary protein sequence. Thus, a second approach is to use these 'signals' to predict specific cell locations. Previous research claimed that proteins which target the secretory pathway, mitochondria and chloroplasts, normally depend on the following N-terminal presequences or targeting peptides:

- secretory pathway signal peptide (SPs): They control the entry of all proteins to the secretory pathway, both in eukaryotes and prokaryotes [25]. SPs generally consist of three regions: a positively charged n-region, a hydrophobic h-region, and a polar c-region leading up to the signal peptidase cleavage site. The most well-conserved motif of SPs is the presence of a small and neutral amino acid at positions -3 and -1 relative to the cleavage site [11].

- mitochondrial targeting peptides (mTPs): In mTPs, amino acids R, A and S are over-represented while negatively charged amino acid residues (D and E) are rare [11].

- chloroplast transit peptides (cTPs): cTPs from different proteins show a wide variety in length and sequence. Still, cTPs has a few distinguishing features such as they tend to be rich in hydroxylated residues and have a low content of acidic residues [10].

Based on these discriminating features for different signals, several sub-cellular localization predictors have been developed, each identifying one particular targeting peptide. For example, SignalP [25] and ChloroP [10] each identified signal peptides using neural networks trained on separate sets of prokaryotic and eukaryotic sequences.

In 2000, TargetP [11] was developed by the same research group that created SignalP and ChloroP. Basically, they integrated and extended their previous efforts to present a predictor that could be used to differentiate four sub-cellular localizations (chloroplast, mitochondrial, extracellular and "other") based on the predicted presence of any of the N-terminal presequences (cTP for chloroplast, mTP for mitochondia, SP for extracellular or no signals for them). It was designed for two categories: plant and non-plant. When we submitted the example sequence (Figure 1.1) to TargetP, it produced the following results:

Sequence Name	mTP	SP	other	Location	RC
test	.017	.952	.063	SP	1

Since it is not a plant sequence, there is no prediction on cTP, instead only scores of mTP, SP are shown. The actual prediction is in column "Location". The scores are not real probabilities so they do not necessarily add up to one. However, the location with the highest score is the most likely one according to TargetP, and the relation between the scores is an indication of how certain the prediction is. TargetP

defines the reliability Class (RC column) as a measure of the size of the difference between the highest and the second highest output scores using a five level scale:

RC =1: diff $\geq$ 0.8

RC =2: .800 $\geq$ diff $\geq$ .600

RC =3: .600 $\geq$ diff $\geq$ .400

RC =4: .400 $\geq$ diff $\geq$ .200

RC =5: .200 $\geq$ diff

In our case, the predicted localization in the 'Location' column is SP (extracellular) and the Reliability Class is 1, indicating the prediction of the query is very confident. However, the assignments of protein N-terminal are usually not reliable using known gene identification methods. As a result, the method is unreliable when the signals are missing or only partially defined. In addition, the known signals are not general enough to create a more complete ontology of protein sub-cellular localizations.

1.2.3 Prediction Based on Similarity Search

Today, given the vast amount of information in protein sequence databases, a third approach is to do a similarity search on the sequence, extract text from homologs and use a classifier on extracted text features. LOCkey [22] is an example of a system that uses this approach. Figure 1.5 illustrates the detailed algorithm:

1. It compiled a data set of proteins whose sub-cellular localization is experimentally known. For each of these proteins, it then extracted keywords from Swiss-Prot.

2. It merged keywords found in homologs and represented the keywords found in proteins of known localization as binary vectors in the 'Trusted Vector Set'.

3. For proteins of unknown localization U, it identified all keywords in U and in homologs of U.

4. It constructed all possible keyword combinations (SUB vectors in Figure 1.5) and compared these to the 'Trusted' vectors. For example, for a protein with 3 keywords, it generated $2^3 - 1 = 7$ SUB vectors: 111, 110, 101, 011, 100, 010 and 001.

5. It found the best matching vector based on entropy criteria methods for inferring the sub-cellular localization of the query sequence.

LOCkey selected all proteins with unambiguously annotated sub-cellular localizations in Swiss-Prot release 40, i.e., it excluded sequences annotated as 'POSSIBLE', 'PROBABLE', 'SPECIFIC PERIODS' or 'BY SIMILARITY', and proteins with multiple annotations of localizations. It handles prokaryotic and eukaryotic proteins together and supports 10 different localizations. As a result, it was applied to annotate five entirely sequenced proteomes including *Saccharomyces cerevisiae (yeast)*. Unfortunately, it has the problem of low coverage, since many proteins lack relevant functional information that maps to the trusted vectors. Since there is no publicly available LOCkey, we could not do our experiment with the example sequence (Figure 1.1). However, the authors shared their dataset with us and in Chapter 5 we compare the accuracy of a PA-SUB classifier trained on this dataset with their published results. PA-SUB has an accuracy almost 10% higher.

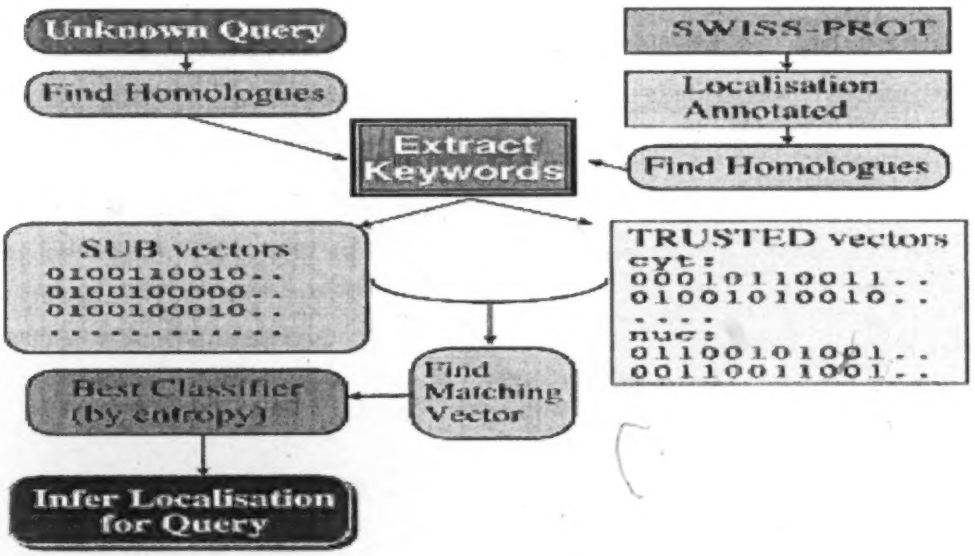

Figure 1.5: LOCkey algorithm.

1.2.4 Combination System

Some of the current sub-cellular localization systems take an integrative approach by combining a variety of individual predictors. The most robust one was developed in the early 90s and is called 'PSORT' [17]. It can be used to predict sub-cellular

localizations for both prokaryotic and eukaryotic proteins. It uses many different individual modules in generating an overall prediction. A query protein undergoes each of the analyses seperately and the results are combined by an expert system. All of the modules in PSORT are binary predictors designed to predict one location only. For example, it employs a very simple method to recognize mitochondrial targeting signals: the discriminant analysis (called "MITDISC") whose variables are the amino acid composition of the N-terminal 20 residues [17]. Certain modules even use more than one technique to analyze the corresponding location. For instance, PSORT involves two steps in recognizing signal sequences. First, it predicts the presence of signal sequences by McGeoch's method [19] modifed in [17]. However, this algorithm does not reveal the location of the cleavage site, which is important to biologists. Therefore, it applies von Heijne's method [30] of signal sequence recognition for the detection of signal-anchor sequences and reports the position of possible cleavage site. In the mid-90s, PSORT's expert system was replaced with a standard classification algorithm (k nearest neighbors), which makes it possible to train the program with a user's own set of data. Using the example sequence (Figure 1.1) for analysis, the PSORT output page has two parts: First is the output of each individual predictor. For example, von Heijne's method shows there is a cleavable signal peptide (1 to 23) for our query sequence, which suggests that the sequence targets the secretory pathway (extracellular). The second part computes and presents the overall prediction results of the combination algorithm, returning both the predicted location site and its probablity. Here is the output for our query sequence:

$\gg$ 44.4%: extracellular, including cell wall

$\gg$ 22.2%: mitochondrial

$\gg$ 11.1%: cytoplasmic

$\gg$ 11.1%: vacuolar

$\gg$ 11.1%: nuclear

$\gg$ prediction for QUERY is extracellular.

More recently, in 2003, PSORT-B [13], an updated version of PSORT, was developed. It specializes in predicting the sub-cellular localization of Gram-negative bacteria, using an ontology with five locations. It examines a given protein sequence for amino acid composition, similarity to proteins of known localization, presence of

a signal peptide, transmembrane alpha-helices and motifs corresponding to specific localizations. It constructs a Bayesian Network to generate a final probabilty for each localization site. PSORT-B is available on the web. However, it only supports Gram-negative bacteria. Therefore, we did not submit the example animal sequence. However, Chapter 5 contains a comparison of PSORT-B to PA-SUB and shows that PA-SUB has higher overall accuracy.

1.2.5 Limitations of Existing Systems

There are two limitations to the work done so far. The first is the limited accuracy of the predictors, especially for some organelles. For example, with amino acid information alone, we cannot produce reliable and accurate predictions. NNPSL achieved only 66% prediction accuracy in prokaryotes. It is also a problem when predictions are more successful for some localization sites than for others. In LOCkey, extracellular, nuclear and mitochondrial proteins could be inferred more reliably than cytoplasmic proteins. One reason could be that experimental annotations are less accurate for cytoplasmic proteins in the database. Another reason could be that proteins do in fact move between the cytoplasm and other localizations.

The second limitation is the coverage. The term *coverage* can be used in two ways. First, given a training/test set, coverage can be defined as the percentage of sequences for which a prediction is made. For instance, the LOCkey dataset consisted of 3146 labeled sequences from Swiss-Prot. A LOCkey classifier obtained an accuracy of 87% on a subset of 1161 of these sequences, which is 37% coverage, i.e. it did not make predictions for the remaining 63% proteins. A second way to measure coverage is to measure the percentage of all the sequences from a given organism that can be classified by a predictor. This is more important measure for high-throughput prediction of sub-cellular localization for newly sequenced organisms.

There are three ways that coverage is limited in current tools. First, the set of cellular regions may be limited (for example, just membranes or just a few organelles). Second, a specific kind of organism may be required (for example, just Gram-negative bacteria). Third, they may only have been evaluated on a limited number of protein sequences. Table 1.3 provides a complete list of predictors and gives a measure of accuracy and coverage.

Table 1.3: Accuracies and representative coverage of current sub-cellular localization predictors

System	Year	Accuracy	Coverage	Technique
PSORT-B [13]	2003	74.8%	1443 Gram-negative	combination of other systems
LOCkey [22]	2002	87% (test set) and 8000 unverified predictions	37% of 3146 test proteins from Swiss-Prot	keyword-vector based classifer
SubLoc [16]	2001	91.4% (prokaryotic) and 79.4% (eukaryotic)	997 prokaryotic in 3 classes and 2427 eukaryotic proteins in 4 classes subset of Swiss-Prot with verified localizations	SVM on amino acid compositions
TargetP [11]	2000	85% (plant) and 90% (non-plant)	940 plant (redundancy reduced Arabidopsis chr 2,4) and 2738 non-plant (Ensembl human proteins)	based on N-terminal amino acid sequence
SignalP [25]	1997	68.0% in Human, 70.2% in Eukaryote, 83.7% in E.coli, 79.3% in Gram- and 67.9% in Gram+	667 human, 1831 Eukaryotes, 224 E.coli, 452 Gram- and 205 Gram+ sequences	signal peptide detection
ProtLock [5]	1997	75% (test set)	200 proteins from Swiss-Prot	amino acid composition
NNPSL [3]	1997	66% (eukaryotes excluding plant) and 81% (prokaryotes)	3420 proteins from Swiss-Prot	amino acid composition
PSORT(II) [15]	1997	86% (E.coli) and 60% (yeast)	336 E.coli in 8 classes and 1484 yeast proteins in 10 classes	k-nearest neighbor
PSORT(II) [14]	1996	81% (E.coli) and 55% (Yeast)	336 E.coli in 8 classes and 1484 yeast proteins in 10 classes	probabilistic classification tree
PSORT(I) [17]	1992	66% of the training data and 59% of the testing data	1484 Yeast sequences	expert system

1.3 Research Goal

The goal of the research described in this dissertation is to overcome the two limitations (accuracy and coverage) of current sub-cellular localization prediction tehniques. This dissertation describes a sub-cellular localization prediction technique that makes the following scientific contributions:

1. This technique makes the most accurate sub-cellular localization predictions over the broadest range of organisms (animals, plants, fungi, Gram-negative bacteria and Gram-positive bacteria) of all published sub-cellular localization prediction techniques.

2. This technique is publicly available as a high-throughput web-based tool in Proteome Analyst [28].

1.4 Summary

This chapter has introduced and motivated the protein sub-cellular localization problem in molecular biology. It has also discussed the state-of-the-art systems produced by other research groups and talked about two major limitations they share: accuracy and coverage. Finally, it stated the goal of our research.

Chapter 2

The Prediction Process

This chapter introduces the software architecture of our proteome anlayst sub-cellular localization prediction system (PA-SUB). It includes a basic description of how a machine learning algorithm produces a classifier/predictor. It also outlines how the constructed classifier is used to make a prediction for an unknown query protein sequence. This chapter also describes the evaluation criteria that was used for PA-SUB.

2.1 A 1-Nearest-Neighbor Prediction

Today, given the vast amount of information in DNA and protein sequence databases, some researchers claim that the chance of finding an annotated sequence that is homologous to a new protein sequence is already approaching 90% for bacteria and is increasing rapidly for eukaryotes [9]. For example, the Swiss-Prot 41 [4] has annotated entries for 80% of the Escherichia coli K-12 bacteria and 75% of the mouse proteome. The goal of our research is to infer sub-cellular localization from the annotated description of similar proteins in these databases.

Specifically, PA-SUB uses Swiss-Prot 41, which contains 122,564 annotated sequences. One of the fields in the Swiss-Prot database directly contains the sub-cellular localization of the protein. We could use the contents of this field for the best homolog to predict the sub-cellular localization of a query sequence. Such a prediction corresponds to 1-Nearest-Neighbor prediction [8]. However, most proteins in the database have no information in this field. In fact, only 55% of the Swiss-Prot entries have a non-empty sub-cellular field and only 37% have an *unambiguous* entry, i.e., an entry that a unique location. As a result, coverage would be poor if a prediction was only made directly from this field of the top homolog. For-

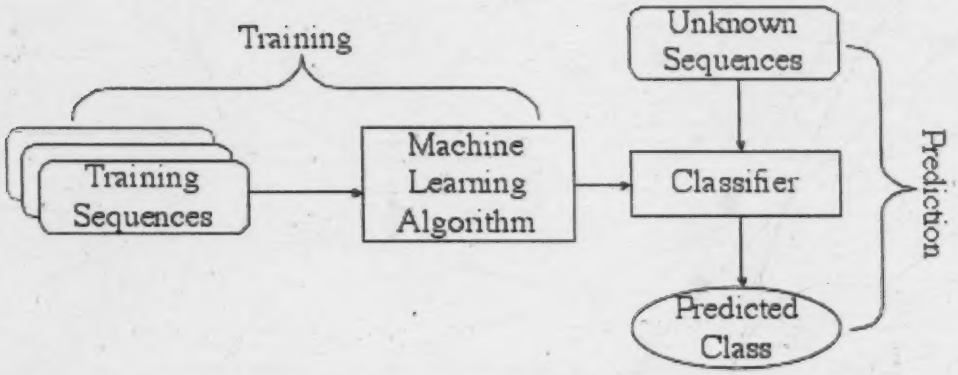

Figure 2.1: The training and predicting phrases of classification in PA-SUB

tunately, we can usually extract some information from other fields and use more than one homolog. These fields can help us make a prediction if we can analyze them effectively.

2.2 Building a Classifier

PA-SUB is based on analyzing information from many fields of several homologs. It uses machine learning classification-based techniques to identify protein sub-cellular localization. In general, it involves a two-step process: training (learning) and prediction. In the training step, a classifier is built using a machine learning algorithm that analyzes a set of training sequences, each tagged by a known sub-cellular localization label. In the prediction step, the generated classifier is used to predict the sub-cellular localization label of an unknown query sequence. Figure 2.1 shows these two steps.

A common technique for building a classifier is to apply a machine learning algorithm to a set of labeled training items to produce a classifier. In general, a machine learning algorithm requires features to be associated with each training item. The value of features (often coded as '0' or '1' for every binary f_i) in each training item, along with the known label (the sub-cellular localization class) for each item, are used to build a classifier.

In our case, each training item consists of a primary protein sequence and the sub-cellular localization label assigned by an expert. For each sequence, we identified

Figure 2.2: PA-SUB builds a classifier using machine learning (ML) algorithm.

a set of feature values, where a feature needed by the classifier is a word or a phrase found in the Swiss-Prot database entries of homologs to the protein in question. We call such a word or phrase a **token**. For example, the word 'zinc' may appear in a field of a homolog. Therefore, if the token 'zinc' is present, the value of f_{zinc}, is '1', whereas if the token 'zinc' is absent, the value of f_{zinc} is '0'. Tokens are not provided in the training items, instead they are computed seperately and automatically by extracting them from homologs. Chapter 4 gives a detailed description of the feature extraction strategy in PA-SUB.

Currently, PA-SUB employs Naïve Bayes (described in Chapter 3) as its default machine learning algorithm, since it has the ability to provide both high prediction accuracies and *transparency* to the users compared with other techniques, such as neural networks. In the context of PA-SUB, *transparency* is the ability to provide formally sound and intuitively simple reasons for each prediction. Explanation is an important factor in getting users to trust predictors. The explanation facilities of PA-SUB are described in [27].

PA-SUB uses labeled training sequences to build a simple Naïve Bayes classifier using the following basic steps:

1. Each labeled training item consists of a primary sequence and a label from the ontology of the classifier being built.

2. The primary sequence of the training sequence is *PSI-BLAST*ed against the Swiss-Prot database. PA-SUB then selects a set of "best" homologous sequences.

3. Values for a set of features are computed by extracting text from the Swiss-Prot records of the best homologs.

4. A set of *sufficient statistics*, $c^+_{i,j}$ and $c^-_{i,j}$, are computed for the set of training instances, where $c^+_{i,j}$ is the number of training sequences that were labeled by

label j and the value of f_i is 1, and $c^-_{i,j}$, is the number of training sequences that were labeled by label j and the value of f_i is 0.

5. A Naïve Bayes classifier is built using the sufficient statistics.

Figure 2.2 illustrates the way we build a machine learning classifier from raw protein primary sequences. PA-SUB actually uses five different classifiers. These are seperate classifiers for: animal, plant, fungi, Gram-positive bacteria and Gram-negative bacteria, since the cell ontology is different for each of these categories of organisms as described in Chapter 1. Other kinds of classifiers were also built (k Nearest Neighbors, Artificial Neural Nets, Tree-augmented Naïve Bayes and Support Vector Machines) during this research. Chapter 3 describes these classifiers and compares their performance to the performance of a Naïve Bayes classifier. Chapter 4 discusses various feature extraction strategies and provides experimental results that compare them.

2.3 Using a Classifier for Prediction

Once built, a classifier takes a list of query sequences with unknown sub-cellular localizations and uses the value of features in homologs of the query sequences to predict their labels. It is actually a *five-step prediction process*:

1. The user-provided category is used to select one of five pre-built Naïve Bayes classifiers: animal, plant, fungi, Gram-positive bacteria and Gram-negative bacteria.

2. The primary sequence of the query protein is *PSI-BLASTed* against the Swiss-Prot database to produce a set of homologous sequences.

3. Values for a set of features are computed by extracting text from the Swiss-Prot records of the best homologs in the same way as when the classifier was built.

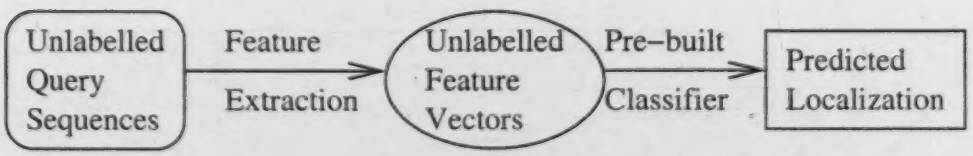

Figure 2.3: PA-SUB makes a prediction using pre-built classifier.

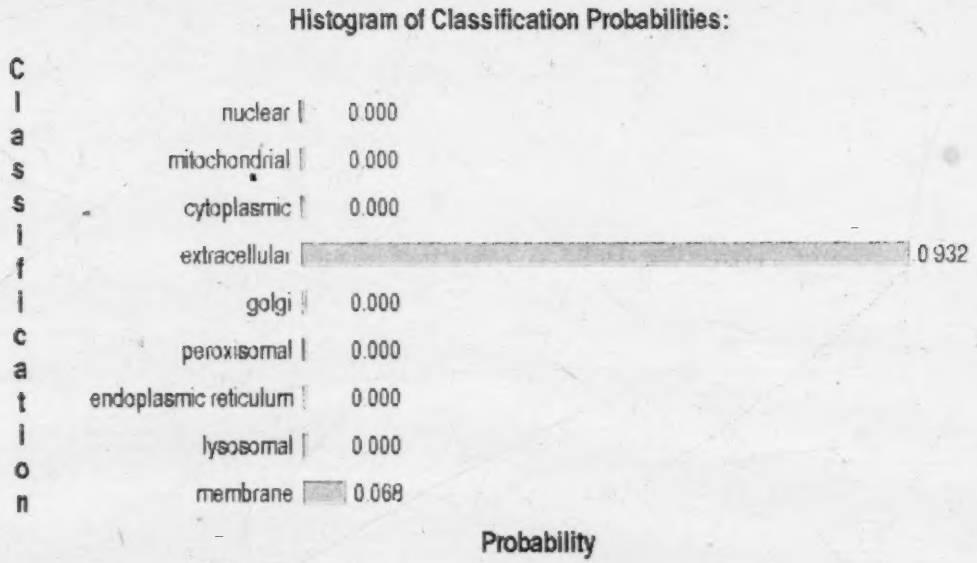

Figure 2.4: PA-SUB predicts location "extracellular" with probability .932 for the example sequence (Figure 1.1) using a pre-built animal classifier.

4. The features are used by the appropriate classifier to compute the probabilities of each of the labels for the ontology of the classifier chosen in Step 1.

5. The user can view the prediction and an explanation in a graphical format [27].

Figure 2.3 shows the prediction for the example sequence (Figure 1.1) as given by PA-SUB. PA-SUB predicts "extracellular" with probability .932. Details of feature extraction step shown in Figure 2.3 is discussed in Chapter 4.

2.4 Classifier Evaluation

To compare classifiers, it is important to define a precise evaluation criteria. We will use a standard machine learning technique called *5-fold cross validation* [20]. Each set of labeled training instances is "randomly" partitioned into five groups of training instances ($G_1 \cdots G_5$), while keeping the number of training instances with each label approximately the same in each training group. Then, five different classifiers are constructed ($C_1 \cdots C_5$), where C_i uses all of the training instances from all of the groups except G_i as test data. Next, a **confusion matrix** [29] is computed for each of the five classifiers, C_i using the sequences in group G_i (the ones

Table 2.1: Confusion matrix for the Gram-negative bacteria classifier, trained on Swiss-Prot data. The abbreviations of ontological labels are shown in Table 1.2. Column no_pred(iction) indicates the number of the sequences that do not have any homologs in the PSI-BLAST search.

Prd⇒ Obs⇓	cyt	ext	per	inn	wal	out	no_pred	SUM^{obs}	recall
cyt	**1775**	21	26	12	0	4	23	1861	.954
ext	3	**216**	7	0	2	1	24	253	.854
per	5	14	**333**	5	0	6	22	385	.865
inn	5	1	4	**411**	0	0	11	432	.951
wal	0	1	0	0	**43**	1	1	46	.935
out	0	4	2	2	0	**181**	8	197	.919
SUM^{prd}	1788	257	372	430	45	193	89	3174	
precision	.993	.840	.895	.956	.956	.938			**.932**

not used in its training) as test data. The final confusion matrix is then computed by summing the entries in all of five confusion matrices.

Table 2.1 shows the confusion matrix for a Gram-negative (GN) bacteria classifier trained on Swiss-Prot data. Each entry in the table represents the number of sequences in the test set whose actual label is the row label and whose predicted label is the column label. For example, the number of sequences with actual label inn(er membrane) that were incorrectly predicted as ext(racellular) is 1. The no_pred(iction) column indicates the number of test sequences that could not be predicted by PA-SUB where no homologs are found by the PSI-BLAST search. For instance, 23 of the cytoplasmic test proteins have no homologs, so no sub-cellular localization prediction could be made by PA-SUB. The SUM^{obs} column indicates the number of test sequences whose actual label is specified by the row label. The SUM^{prd} row indicates the number of test sequences whose predicted label is specified by the column label.

Various statistics can be computed from the confusion matrix to evaluate the effectiveness of a classifier. In this dissertation we will use three standard statistics. Give a confusion matrix M and a set of labels $\{L_i\}$, the standard definitions [29] of these statistics are as follows. The *precision* for each label L_i is:

$$P_i = M_{ii} / \sum_{k=1}^{n-1} M_{ki} = M_{ii} / SUM_i^{prd} \tag{2.1}$$

The *recall* for each label L_i is R_i defined by:

$$R_i = M_{ii} / \sum_{j=1}^{n} M_{ij} = M_{ii} / SUM_i^{obs} \tag{2.2}$$

The *accuracy* of the classifier, A, is defined by:

$$A = \sum_{i=1}^{n-1} M_{ii} / \sum_{k=1}^{n-1} \sum_{j=1}^{n} M_{kj} = \sum_{i=1}^{n-1} M_{ii} / sum \tag{2.3}$$

The coverage of the classifier, C, is defined by:

$$C = \frac{sum - sum_{no\_prediction}}{sum} \tag{2.4}$$

where *sum* is the total number of test sequences (3174 in Table2.1) and n is the total number of different labels including the 'no_prediction' column (7 different labels in Table 2.1). For example, the precision of label inner membrane is $P_{(inn)} = 411/430 = .956$, the recall of label inner membrane is $R_{(inn)} = 411/432 = .951$, the accuracy of this classifier is $A = (1775 + 216 + 333 + 411 + 43 + 181)/3174 = .932$ and the coverage of the classifier is $C = (3174 - 82)/3174 = .974$. The accuracy can be viewed as the overall recall, since it can be expressed as a weighted average of the recalls of all of the labels, where each weight is the number of test sequences with each actual label (SUM_i^{obs}):

$$\bar{R} = \frac{\sum_{i=1}^{n-1} SUM_i^{obs} R_i}{sum} = \frac{\sum_{i=1}^{n-1} SUM_i^{obs} (\frac{M_{ii}}{SUM_i^{obs}})}{sum} = \frac{\sum_{i=1}^{n-1} M_{ii}}{sum} = A \tag{2.5}$$

2.5 Summary

In this chapter we have introduced the software architecture of the sub-cellular prediction server (PA-SUB). We have described how a classifier can be built and how a prediction can be made using the pre-bulit classifier. In addition, we presented statistics for evaluating a classifier precisely. Chapter 3 presents more details on the machine learning algorithms.

Chapter 3

Machine Learning and Classification

This chapter will describe the five established machine learning algorithms we used to build the PA-SUB classifiers, including k Nearest Neighbors (kNN), Naïve Bayes (NB), Tree-augmented Naïve Bayes (TAN), Artificial Neural Net (ANN) and Support Vector Machine (SVM). In addition, it will present the reasons why we finally chose NB as our default classification technique.

3.1 Classification: A Two-Step Process

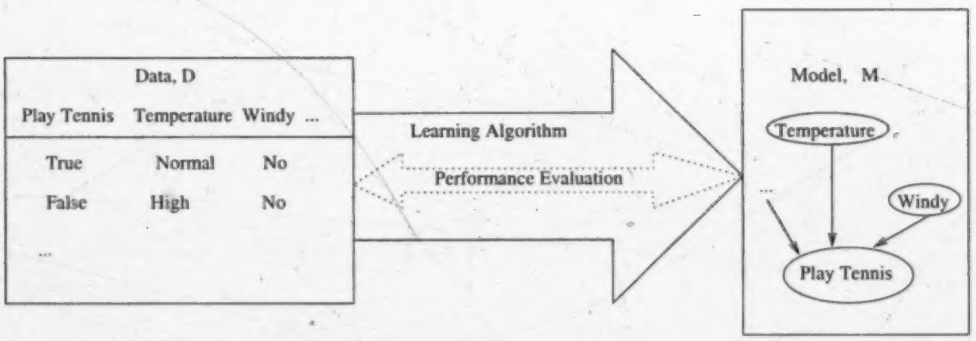

Figure 3.1: A Two-Step process: learning and evaluation.

Figure 2.1 shows the software architecture of PA-SUB. More generally, every classification process involves the basic components illustrated in Figure 3.1: A learning algorithm defines the actual learning process. It defines how to build an optimal model M from data D according to a certain evaluation function, which is important as it is used to determine how close the model is to the original objects. Models are

often used by learning algorithms to represent knowledge. A good model allows the knowledge to be stored and used efficiently. Bayesian nets, neural nets and support vector machines are the models widely used in machine learning [20]. They will be explained in detail in this chapter.

3.2 k Nearest Neighbors

The k nearest neighbors classifier [8] stores the complete training instances. Each instance $x_1, \cdots, x_n, x_i \in R^d$ is represented by its attribute vector $\langle A_1 = a_1, \cdots, A_d = a_d \rangle$. As an example, consider Figure 3.1, A_1 might be the $Temperature$, A_2 might be $Windy$ and $x_1 = \langle normal, no \rangle$. The distance $d(x_i, x_j)$ of two instances x_i and x_j, a measure for their similarity, is usually defined by their vector distance. The most popular distance measure is **Euclidean distance**, which is defined as:

$$d(x_i, x_j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \cdots + |x_{id} - x_{jd}|^2} \tag{3.1}$$

To classify a new instance from the set of stored instances, the k instances most similar to the query object are determined, for some constant k, e.g., $k = 1$ or $k = 3$. The query object is then classified by chosing the majority class among the k closest examples. For instance, if given a new instance $\langle Temperature = normal, Windy = yes \rangle$, the most similar instance shown in Figure 3.1 is $\langle normal, no \rangle$, since for that instance, $d(\langle normal, yes \rangle, \langle normal, no \rangle) = 1$ and for the other training instance $d(\langle normal, yes \rangle, \langle high, no \rangle) = 0$. Consequently, a 1NN classifier will use the class "Play Tennis = true" of the training instance $\langle normal, no \rangle$ as the predicted class for the query instance $\langle Temperature = normal, Windy = yes \rangle$.

For PA-SUB, we used training sequences from the Swiss-Prot protein database. A sequence similarity search is used to determine the similarity among them. Therefore, for a query protein, we perform the following two steps:

- First, we selected k most similar proteins in the database using PSI-BLAST.
- Second, we analyzed the SCELL[1] field in those homologs to determine the sub-cellular localization for the query sequence.

[1] We explain the details in Chapter 4

Table 3.1: Confusion matrix for the 1NN classifier, trained on GN bacteria data from Swiss-Prot.

Prd⇒ Obs⇓	cyt	ext	per	inn	cew	out	no_pred	SUM^{obs}	recall
cyt	**1692**	0	4	1	0	0	164	1846	.917
ext	1	**183**	6	0	0	0	63	249	.740
per	4	3	**301**	3	0	3	71	366	.822
inn	0	1	2	**370**	0	0	59	430	.860
wal	0	0	0	0	**42**	1	3	46	.913
out	0	0	2	0	0	**168**	27	193	.870
SUM^{prd}	1697	187	315	374	42	172	387	3174	
precision	.997	.979	.956	.989	1.00	.978			**.868**

Table 3.1 shows the results when we selected the single most similar sequence in the database, i.e., $k = 1$. We also evaluated kNN classifiers for $k = 3$ and $k = 5$; the results are shown in Chapter 5.

3.3 Naïve Bayes

In this dissertation, we consider the Naïve Bayes classifier [20] as the system default technique because of its high accuracy and transparency [27]. It is a type of Bayesian network (BN) classifier. These classifiers are usually applied to learning tasks where each instance x is described by a conjunction of attribute values and where the target function $f(x)$ can take on any value from some finite set C. For example, C represents *PlayTennis* in Figure 3.2, and has two real values: $C_1 = true$ or $C_2 = false$.

Given a new instance, described by the tuple of attribute values $\langle f_1, \cdots, f_n \rangle$, e.g., F_1 might be $Temperature = normal$, F_2 might be $Windy = no$, the classifier is asked to predict the target class. To do that, the classifier must learn, from training data, the conditional probability of each attribute F_i given the class label C. For example, the left most table in Figure 3.2 claims the probability of $Temperature = normal$ given *Play Tennis = true*, *P(Temperature=normal | Play Tennis = true)* is 0.90. These tables are called *conditional probability tables* (CPTs). By using the probabilities from CPTs, probabilistic classifiers like BN will typically compute posterior probabilities for each $P_j = P(C = c_j | F_1 = f_1, \cdots, F_n = f_n)$ and then return the class label $C^* = argmax_j P_j$.

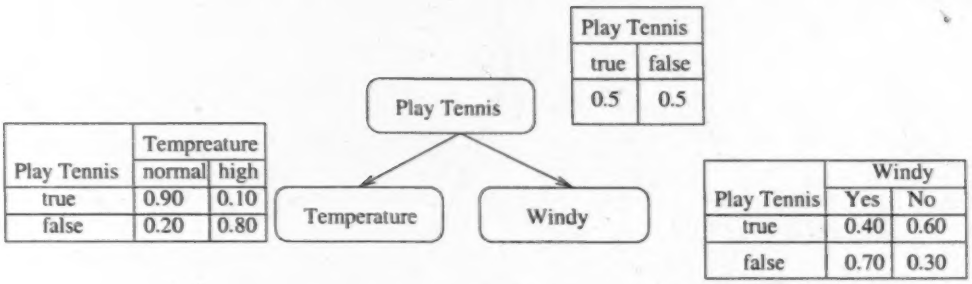
Play Tennis
Play Tennis
Temperature
Windy
Play Tennis | true | false
0.5 | 0.5
Play Tennis | Tempreature: normal | high
true | 0.90 | 0.10
false | 0.20 | 0.80
Play Tennis | Windy: Yes | No
true | 0.40 | 0.60
false | 0.70 | 0.30

Figure 3.2: An example of a simple Naïve Bayes classifier.

Naïve Bayes is one of the most effective BN classifiers with a simple structure, based on the assumption that attributes are independent given class labels. As illustrated in Figure 3.3, the node for class labels serves as the root of the tree, all the attributes are the child nodes of the root, and no links to siblings exist for each child node indicating every attribute is independent from the rest of the attributes. In other words, the assumption is that given the target class of the instance, the probability of observing the conjunction $F_1, \cdots, F_n$ is just the product of the probabilities for the individual attributes $P(F_1 = f_1, \cdots, F_n = f_n | C = c_j) = \prod_i P(F_i = f_i | C = c_j)$. Given this assumption, we can derive a fomula using Bayes Rule:

$$\begin{aligned} P_j &= P(C = c_j | F_1 = f_1, \cdots, F_n = f_n) \\ &= \frac{P(F_1 = f_1, \cdots, F_n = f_n | C = c_j) P(C = c_j)}{P(F_1 = f_1, \cdots, F_n = f_n)} \\ &= \frac{P(C = c_j) \prod_i P(F_i = f_i | C = c_j)}{P(F_1 = f_1, \cdots, F_n = f_n)} \qquad (3.2) \\ &= \propto * P(C = c_j) * \prod_i P(F_i = f_i | C = c_j) \qquad (3.3) \end{aligned}$$

where $\propto = \frac{1}{P(F_1 = f_1, \cdots, F_n = f_n)}$ is identical for all different classes $C = c_j$, and is used to make the probabilities sum to 1. We can define an unnormalized value, $Q(C = c_j)$

$$Q(C = c_j) = P(C = c_j) \prod_i P(F_i = f_i | C = c_j) \qquad (3.4)$$

$$C^* = argmax_j Q(C = c_j) \qquad (3.5)$$

$$C^* = P(C = c_j) \prod_i P(F_i = f_i | C = c_j) \qquad (3.6)$$

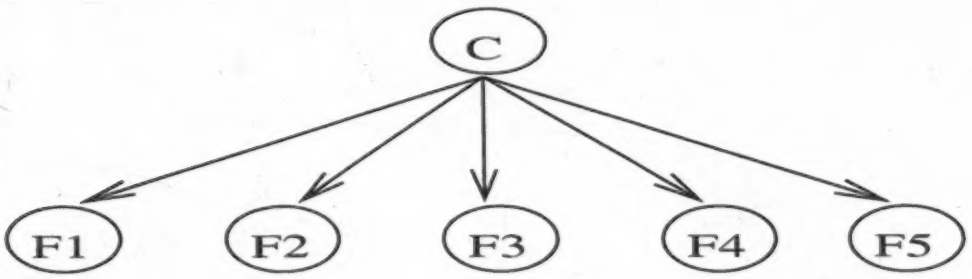

Figure 3.3: An example of Naïve Bayes classifier structure. Here, nodes are variables and links between nodes represent the causal dependency.

To build a Naïve Bayes classifier involves a learning step in which the various $P(C = c_j)$ and $P(F_i = f_i|C = c_j)$ terms are estimated, based on their frequencies over the training data. To use the classifier for predicting unknown objects, it employs those pre-computed CPTs and applies Formula 3.5 to the new object. For example, if given a new instance $\{Temperature = normal, Windy = yes\}$, it computes

$$\begin{aligned} Q_{PlayTennis=true} &= P(PlayTennis = true) * \\ &\quad P(Temperature = normal|PlayTennis = true) * \\ &\quad P(Windy = yes|PlayTennis = true) \\ &= 0.5 * 0.9 * 0.4 = 0.18 \end{aligned}$$

and $Q_{PlayTennis=false} = 0.07$, therefore, the target class is to 'play tennis'. The probabilities are actually $P_{PlayTennis=true} = \frac{.18}{.18+.07} = .72$ and $P_{PlayTennis=false} = \frac{.07}{.18+.07} = .28$. We already showed the results of using this technique to classify the GN bacteria in Table 2.1.

3.4 Tree-augmented Naïve Bayes

An obvious enhancement to a NB classifier is to improve its structure by adding links among the attributes. One such successful extension is the tree-augmented Naïve Bayes (TAN) [12], which uses Chow-Liu's algorithm [6] to add links that form a tree among the attributes. A graphical illustration is shown in Figure 3.4.
Below is the algorithm we used to construct the TAN tree, taken from [12]:

1. Compute the *Conditional Mutual Information* $I_p(F_i = f_i, F_j = f_j|C = c_k)$ between every two features F_i and F_j, where all the probablities like $P(F_i =$

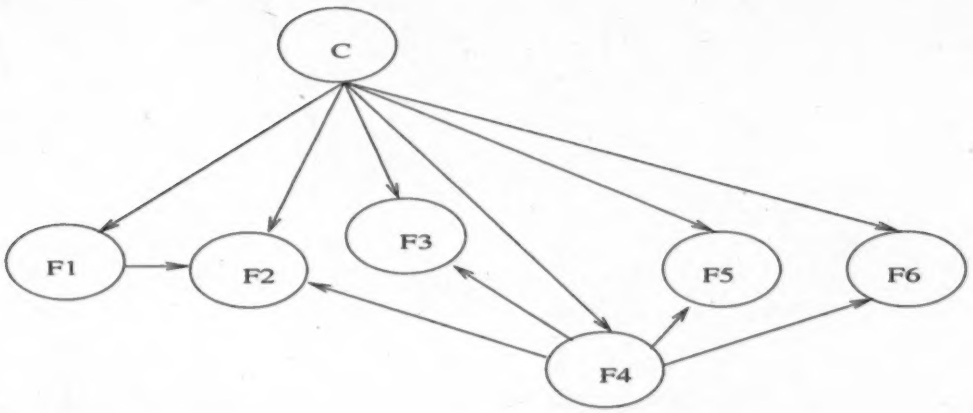

Figure 3.4: An example of Tree-augmented Naïve Bayes classifier. Under this model, we allow some additional edges between attributes for representing simple correlations between the attributes.

$f_i, F_j = f_j|C = c_k)$ are estimated, based on their frequencies over the training data.

$$I_p(F_i = f_i, F_j = f_j|C = c_k) = \sum_{f_i,f_j,c_k} P(F_i = f_i, F_j = f_j, C = c_k) log \frac{P(F_i = f_i, F_j = f_j|C = c_k)}{P(F_i = f_i|C = c_k)P(F_j = f_j|C = c_k)}$$

2. Construct a complete undirected graph over feature nodes, where the features are the nodes and the $I_p(F_i = f_i, F_j = f_j|C = c_k)$ values are the edge weights between features F_i and F_j.

3. Generate a maximum weighted spanning tree from the graph.

4. Choose the feature with the highest information content (described in Chapter 4) to be the root node, forming a directed tree.

5. Add the classification node as a parent to all other nodes.

Once both the tree structure and related CPTs have been learned, we can use the model to classify new instances. Given a data record with attribute values $\langle F_1 = f_1, \cdots, F_n = f_n \rangle$:

$$Q(C = c_j) = P(C = c_j) \prod_i P(F_i = f_i|parent(F_i), C = c_j)$$
$$= P(C = c_j) \prod_i P(F_i = f_i, parent(F_i), C = c_j)/P(parent(F_i), C = c_j)$$

Table 3.2: Confusion matrix for the TAN classifier, trained on GN bacteria data from Swiss-Prot.

Prd⇒ Obs⇓	cyt	ext	per	inn	cew	out	no_pred	SUM^{obs}	recall
cyt	**1779**	16	28	14	0	4	20	1861	.956
ext	3	**215**	8	0	0	3	24	253	.850
per	7	13	**336**	5	0	6	18	385	.873
inn	4	4	3	**410**	0	0	11	432	.949
wal	0	8	0	0	**35**	2	1	46	.761
out	2	1	2	2	0	**182**	8	197	.924
SUM^{prd}	1795	257	377	431	35	197	82	3174	
precision	.991	.837	.891	.951	1.00	.924			**.932**

where $parent(F_i)$ is the value of feature F_i's parent, and we consider the root node to be: $parent(F_{root}) = \{\}$. It has been shown [12] that TAN produces excellent classification results, usually better than NB. We used TAN to create a GN bacteria classifier whose overall accuracy was .932 (shown in Table 3.2). This is the same result as the Naïve Bayes classifier, but with a different classifier (TAN). For example, TAN is better in precision but worse in recall for cell wall (cew) class compared with the NB results shown in Figure 2.1.

3.5 Artificial Neural Nets

Artificial neural networks provide a practical method for learning real-valued and vector-valued functions over continuous and discrete-valued attributes, in a way that is robust to noise in the training data. The Backpropagation algorithm [20] is the most common network learning method and has been successfully applied to a variety of learning tasks, such as handwriting recognition and robot control.

The networks are composed of many computational elements or nodes (so-called neurons) that are connected in several layers (input, hidden and output) via weights, which are typically adjusted during the training phase to achieve high accuracy. A typical multi-layer neural network is shown in Figure 3.5.

The weights of a neural network are learned during the training phase. The attributes of a training instance are given to the input layer and the weights are automatically adjusted so that the output layer indicates the correct label of the training instance. This process is repeated for all training instances. The classifier

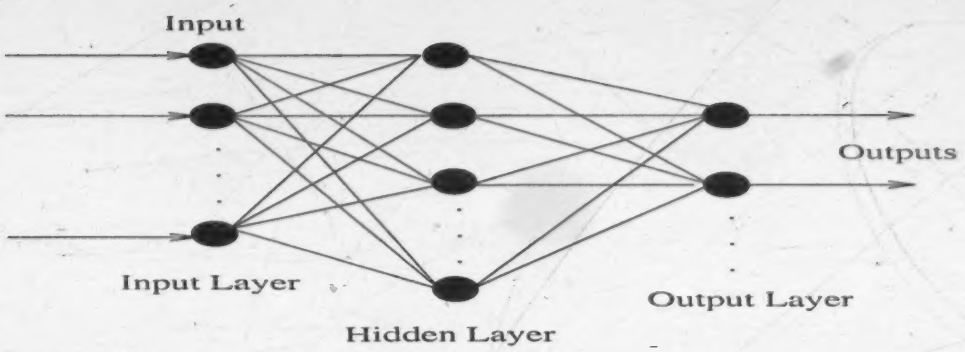

Figure 3.5: An example of a Multi-Layer Neural Network. It has three layers including input layer, hidden layer and output layer. Each • unit above represents a neuron.

is then represented by the neural network with the learned weights.

Figure 3.6 illustrates a simple computation: it receives inputs and computes a new activation level that it sends to output. The computation of the activation level is based on the values of each input $\bar{X} = (x_1, \cdots, x_n)$ and the respective weights $(w_1, \cdots, w_n)$. The computation is split into two components. First is a *linear* component, called the **input function**, $in(\cdot)$, that computes the weighted sum of the unit's input values. Second is a *nonlinear* component called an **activation function**, $g(\cdot)$, that transforms the weighted sum into the final value [26]. More precisely, the input function is $in(\bar{X}) = \sum_{i=1}^{n} x_i w_i$ and the activation function is referred to as a **sigmoid function**: $g(y) = \frac{1}{1+e^{-y}}$. note that its output ranges between 0 and 1, increasing monotonically with its input as shown in Figure 3.6. The final output for this unit is: $o(\bar{X}) = g(in(\bar{X})) = \frac{1}{1+e^{-in(\bar{X})}}$.

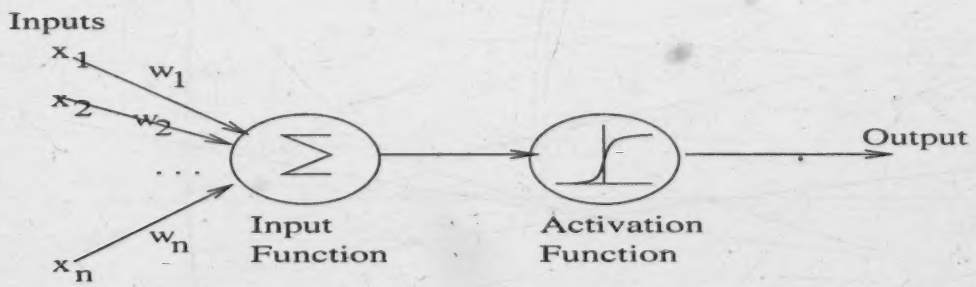

Figure 3.6: A unit.

The steps of learning weights of a single-layer neural network are the followings:

1. initialize weights with random values
2. **foreach** training instance $\bar{X}$
   compute the input to the network as a linear combination of all the inputs to the input layer
   compute the value for the output node using the activation function
   evaluate the error between the expected outcome and the predicted value
3. feed all training instances into the network
4. update the weights of the network

Learning in multi-layer neural networks proceeds the same way as described above: sample inputs are presented to the network, and if the network computes an output that matches the target, nothing is done. If there is an error (a difference between the output and target), then the weights are adjusted to reduce this error [26]. In single-layer networks, this is easy, because there is only one weight between each input and the output. But in multi-layer network, there are many weights connecting each input to an output. The Backpropagation algorithm [20] is used to learn the weights for a multi-layer network, given a network with a fixed set of units and interconnections.

Once the classifier is built, it can be used to classify new instances. To classify a new instance, its attributes are loaded into the nodes in the input layer and the class label is generated by the output layer.

For PA-SUB in particular, we used a three-level ANN, with the number of input nodes equal to the number of features, a single layer of hidden units and a number of output units equal to the number of ontology classes. We started using 10 hidden units; our experiments showed that varying this number had little effect. We repeated the weight update phase 100 times for each instance in the training set to achieve best results.

For comparison purposes, we constructed a confusion matrix for the ANN classifier, built using the same GN bacteria data. The overall accuracy (.956) in Table 3.3 is 2.4% better than the NB classifier and the TAN classifier.

Table 3.3: Confusion matrix for the ANN classifier, trained on GN bacteria data from Swiss-Prot.

Prd⇒ Obs⇓	cyt	ext	per	inn	cew	out	no_pred	SUM^{obs}	recall
cyt	**1815**	10	13	3	0	0	20	1861	.975
ext	1	**225**	3	0	0	0	24	253	.889
per	6	10	**350**	0	0	1	18	385	.909
inn	0	2	2	**417**	0	0	11	432	.965
wal	0	0	0	0	**44**	1	1	46	.957
out	1	3	2	0	0	**183**	8	197	.929
SUM^{prd}	1823	250	370	420	44	185	82	3174	
precision	.996	.900	.946	.993	1.00	.989			**.956**

3.6 Support Vector Machines

A simple SVM basically learns how to classify objects into two classes $\{+1, -1\}$ from a set of labeled training instances $\{\langle x_i, y_i \rangle\}, i = 1, \cdots, l, x_i \in R^d, y_i \in \{+1, -1\}$. The main idea is to find the hyperplane h that separates the training instances by a maximal margin. The margin is defined as the minimal distance of an instance to the hyperplane. The instances that lie closest to the separating hyperplane (both positive and negative instances) are called *support vectors.*

If the training instances are separable by a hyperplane (see Figure 3.7), we could choose functions of the form $f(x) = (w \cdot x) + b$. When $f(x) \geq 0$, we classify x as the positive class; otherwise, we classify x as the negative class.

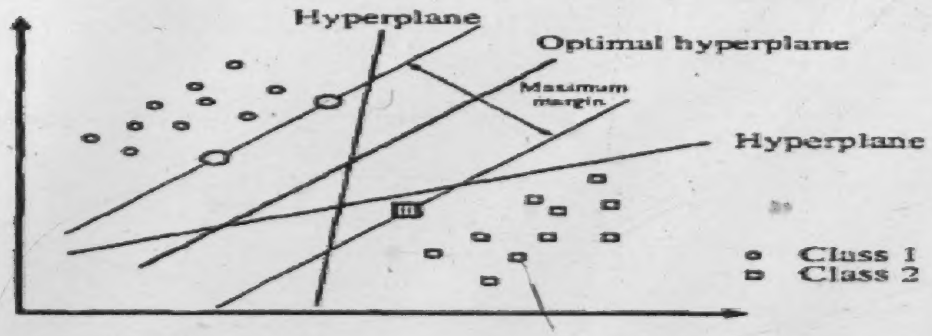

Figure 3.7: A linear classifier is defined by a hyperplane's normal vector w and an offset b, i.e. the decision boundary is $\{x|(w \cdot x) + b = 0\}$. Each of the two halfspaces defined by this hyperplane corresponds to one class, i.e. $f(x) = sign((w \cdot x) + b))$. The objects that are circled are called *support vectors* [21].

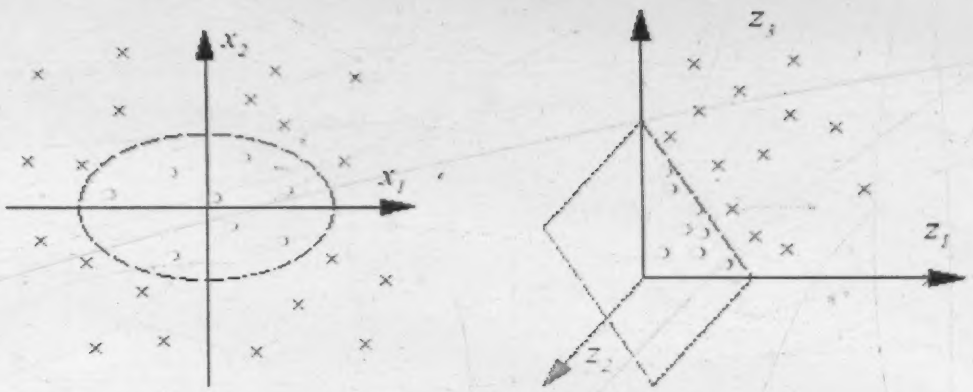

Figure 3.8: A two dimensional classification example. Using the second order monomials $x_1^2, \sqrt{2}x_1x_2, x_2^2$ as features a separation in feature space can be found using a *linear* hyperplane (right). The input space is a non-linear dataset (left) [21].

More generally, SVM allows us to project the original training instances into a higher dimensional feature space via a nonlinear mapping [21]:

$$\Phi : \Re^d \rightarrow F$$

$$x \rightarrow \Phi(x)$$

the data $x_1, \cdots, x_n \in \Re^d$ is mapped into a potentially much higher dimensional feature space F. For a given learning problem one can now consider the same algorithm in F instead of R^d. There are many different mapping functions such as a polynomial function as shown in Figure 3.8. This research has evaluated several mapping functions and finally used a linear function since no significant differences were noted on this particular task. For a multi-class problem, SVM typically divides the problem into disjoint binary classification tasks. As a result, to classify a new object, all binary classifiers are envoked and their decisions are combined to make a final call. Table 3.4 presents the confusion matrix computed by an SVM classifier using the GN datase. The overall accuracy is second only to the ANN classifier.

3.7 Summary

This chapter described the major classification techniques we tried in PA-SUB and presented the results. Table 3.5 shows that the NB accuracy is 6.4% better than the KNN classifier and is tied with the TAN classifier, but is inferior to the ANN and

Table 3.4: Confusion matrix for the SVM classifier, trained on GN bacteria data from Swiss-Prot.

Prd⇒ Obs⇓	cyt	ext	per	inn	cew	out	no_pred	SUM^{obs}	recall
cyt	**1816**	6	15	2	0	2	20	1861	.976
ext	15	**199**	9	0	2	4	24	253	.786
per	21	8	**330**	5	0	3	18	385	.857
inn	7	0	3	**411**	0	0	11	432	.951
wal	0	1	0	0	**43**	1	1	46	.935
out	2	3	3	0	0	**181**	8	197	.919
SUM^{prd}	1861	217	360	418	45	191	82	3174	
precision	.976	.917	.917	.986	.956	.948			**.939**

Table 3.5: A comparison of the statistics of all four machine learning algorithms, built using the same dataset, GN bacteria from Swiss-Prot. The ontological labels are: (cyt)oplasmic, (ext)racellular, (per)iplasmic, (inn)er membrane, cell (wal)l and (out)er membrane.

loc	count	NB		TAN		ANN		SVM		1NN	
		prec	rec	prec	rec	prec	rec	prec	rec	prec	rec
cyt	1861	.993	.954	.991	.956	.996	.975	.976	.976	.997	.917
ext	253	.840	.854	.837	.850	.900	.889	.917	.786	.979	.740
per	385	.895	.865	.891	.873	.946	.909	.917	.857	.956	.822
inn	432	.956	.951	.951	.949	.993	.965	.986	.951	.989	.860
wal	46	.956	.935	1.00	.761	1.00	.957	.956	.935	1.00	.913
out	197	.938	.919	.924	.924	.989	.929	.948	.919	.978	.870
	3174	acc.→	**.932**	acc.→	**.932**	acc.→	**.956**	acc.→	**.939**	acc.→	**.868**

SVM classifiers by 1% - 3% on the GN bacteria data. However, it is very difficult to explain the predictions of ANN and SVM classifiers, which is an important factor in getting users to trust predictors [27]. Therefore, we chose to use NB as the system (PA-SUB) default algorithm. The next chapter discusses the strategies used to extract the best features to be used by the ML algorithms. These strategies are key to gaining high accuracies.

Chapter 4

Feature Extraction and Selection

As introduced in Chapter 2 and 3, protein features are critical in both the process of building a classifier and of predicting the class of an instance, using the learned classifier. This chapter discusses the strategies we used to extract protein features from protein primary sequences and the technique we used to select the most important features among all the potential features in order to build the most accurate classifiers.

4.1 Feature Extraction

4.1.1 Introduction

Feature extraction is a pre-processing step in PA-SUB as shown in Figure 2.2 and 2.3 of Chapter 2. This step maps each sequence to a set of features. Figure 4.1 illustrates the feature extraction process. The sequence is first PSI_BLASTed [2] against the Swiss-Prot database to obtain similar sequences called *homologs*. The Swiss-Prot entry of each of the top homologues is parsed to extract a set of tokens. The union of the tokens for all these top homologs forms the token set for the protein sequence. In this way, we collect all the tokens that appear in the training set and build a fixed set of tokens over all the training sequences. Finally, we identify each sequence with the presence or absence of each token. If no homologs are found during the database search, then the sequence will have no tokens, so no prediction will be made.

We tried many different variations of this feature extraction process to enhance the classifier accuracy. However, we did not vary the Swiss-Prot [4] protein database.

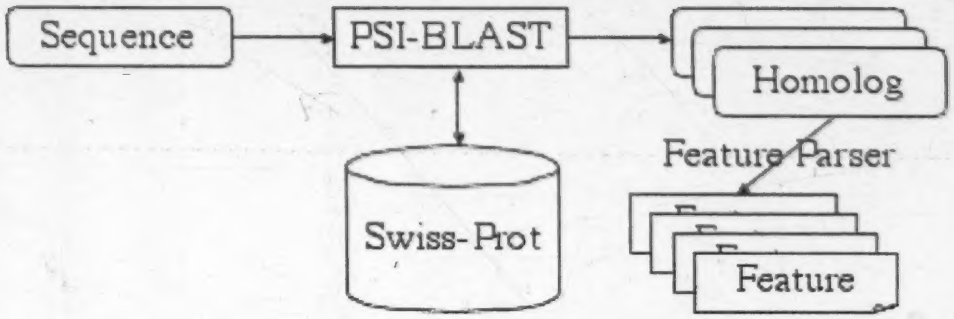

Figure 4.1: Computing features for a protein sequence in PA-SUB

It is a curated protein sequence database with a high level of annotations, a minimal level of redundancy, a high level of integration with other databases, and fewer errors than non-curated sequence databases. The four variables in our feature extraction experiments were:

1. The number of iterations of PSI-BLAST.
2. The number of top homologs selected.
3. The Swiss-Prot fields from which features were selected.
4. The structure of the extracted features.

4.1.2 The Number of PSI-BLAST Iterations

Many functionally and evolutionarily important protein similarities are recognizable only through comparison of three-dimensional structures. When such structures are not available, the protein sequences of amino acids themselves are not informative. Therefore, they must be analyzed by comparative methods against existing databases to obtain similar sequences that might have similar function.

The general approach involves the use of a set of algorithms to compare a query sequence to all the sequences in a specified database. Comparisons are made in a pairwise fashion. Each comparison is given a score reflecting the degree of similarity between the query and the database sequence being compared. The higher the score, the greater the degree of similarity. The similarity is measured and shown by aligning two sequences. There are two types of alignments: global and local. A

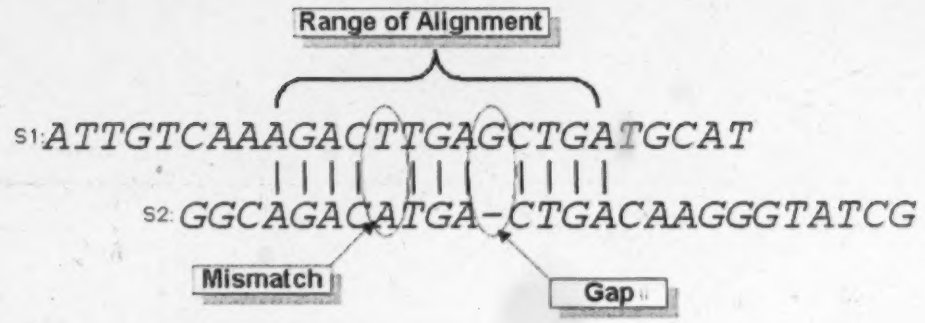

Figure 4.2: An example of sequence alignment. For simplicity, we allow only 4 letters to appear: A(alanine), C(cysteine), G(glycine), and T(threonine). Coincidentally, these also happen to be the letters used for four nucleotides(adenine, cytosine, guanine, thymine) of DNA.

global alignment is an *optimal alignment* (i.e., an alignment of two sequences with the highest possible score) that includes all letters from each sequence, whereas a local alignment is an optimal alignment that includes only the most similar local region or regions. For example, assume we have two sequences s_1 and s_2 and the alignment shown in Figure 4.2. For simplicity, we will count $+2$ when we get an identity in a column of an alignment, -1 when we have a mismatch, and -2 when we insert a gap. So, the similarity score for the range of alignment is $+2+2+2+2-1+2+2+2-2+2+2+2+2=19$.

The BLAST [1] programs (Basic Local Alignment Search Tools) are a set of sequence comparison algorithms introduced in 1990 that are used to search sequence databases for optimal local alignments to a query. PSI-BLAST[2], Position-Specific Iterated BLAST, can do an iterative search in which sequences found in one round of searching are used to build a scoring model for the next round of searching. Thus it uncovers many protein relationships missed by single pass database search methods and has identified relationships that were previously detectable only from information about the three-dimensional structure of the proteins. Note that BLAST equals PSI-BLAST if the latter does only one iteration. For this research, we experimented with the number of PSI-BLAST iterations.

4.1.3 The Number of Homologs to Use

Once a protein sequence is PSI-BLASTed against the Swiss-Prot database, there will be a list of similar protein sequences generated. Therefore, we varied the number of

homologous proteins used to extract features from. No matter how many homologs are selected, all of them must be similar enough to our query sequence. This was evaulated using the *E-value*, an estimate of the probability of discriminating between real and random matches that might occur by chance.

The Expectation value (E-value) is a parameter that describes the number of matches one can "expect" to see just by chance when searching a database of a particular size. Essentially, the E value describes the random background noise that exists for matches between sequences. For example, an E-value of 1 assigned to a hit can be interpreted as meaning that in a database of the current size one might expect to see 1 match with a similar score simply by chance. This means that the lower the E-value (i.e., closer to "0") the more "significant" the match is. However, it is easy to randomly have a short sequence that has a relatively high E-value because the calculation of the E-value also takes into account the length of the query sequence. This is because shorter sequences have a high probability of occuring in the database purely by chance.

We always select the similar sequences with the lowest E-value, but never take any sequences with E-value $>$ .001. The cutoff was first suggested using domain knowledge and then we varied its threshold. A threshold of .001 produced the best results.

4.1.4 Selecting Swiss-Prot Fields

Each annotated protein in Swiss-Prot has many different annotation fields. Each field provides some information about the protein. Example fields are the name, keywords and sub-cellular location. Part of a Swiss-Prot entry is shown in Figure 4.3. Some of these fields are required such as its name, some are optional such as its sub-cellular localization. We have experimented with the following fields to extract features for our sub-cellular classifiers:

- **Sub-cellular localization (SCELL)** field: it has the description, in free text format, of the sub-cellular location of the mature protein. It is an optional field so that we can not count on its presence to make a prediction. For example, "Mitochondrial matrix" is in the SCELL field of protein MPPB_NEUCR as shown in Figure 4.3. Other example contents are "peroxisomal or cytoplasmic" and "predominantly mitochondrial but also membrane associated and cytoplasmic". Note that it is often impossible to extract an exact ontological

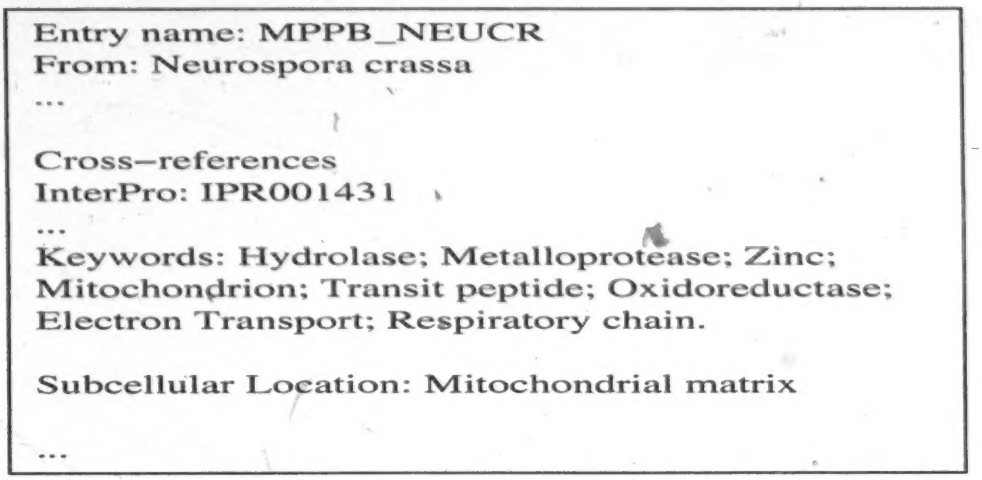

Figure 4.3: Extracting tokens from a Swiss-Prot entry in PA-SUB

label for this field. Therefore, a k nearest neighbor classifier might be hard to use, as the exact label for the neighbor might not exist. As a result, it does not necessarily produce the best accuracy (See Chapter 5).

- **Keywords (KW)** field: it provides information that can be used to generate indexes of the sequence entries based on functional, structural, or other categories. The keywords chosen for each protein serve as a subject reference for the sequence. It is an optional field as well. For example, the Keyword field of the protein in Figure 4.3 contains "Hydrolase; Metalloprotease; Zinc; ...". The keywords are seperated by semi-colons.

- **InterPro (IPR)** field: InterPro is a database of protein families. Most of the proteins in Swiss-Prot have a cross-reference to InterPro. For example, protein MPPB_NEUCR has InterPro number "IPR001431".

We experimented with various combinations of these three fields. For instance, we tried using only one field such as SCELL and tried using all three fields.

4.1.5 Words or Phrases

There are many ways to extract tokens from these fields. For example, we could use each single word as a token. Alternatively, we could consider several adjacent related words as a phrase token. Therefore, we varied the way we parsed the SCELL and KW fields.

Table 4.1: A comparison of the accuracy of some Gram-negative classifiers, which use different homolog selection techniques (number of PSI_BLAST iterations and number of top homologs), different Swiss-Prot fields (KW, IPR and SCELL) and different feature extraction techniques (semi-colon delimited phrases or words for KW, and fixed phrases or words for SCELL).

PSI_BLAST iterations	Top Homologs	KW	IPR	SCELL	Accuracy
1	**3**	**phrase**	**yes**	**phrase**	**.932**
1	3	phrase	yes	no	.922
1	3	phrase	no	phrase	.930
1	3	no	no	phrase	.923
2	3	phrase	yes	phrase	.925
1	2	phrase	yes	phrase	.931
1	4	phrase	yes	phrase	.931
1	3	words	yes	words	.923

- KW field: We tried treating semi-colon delimited phrases like: "Transit peptide" in Figure 4.3 as a single token or alternatively, as two seperate tokens.
- SCELL field: We tried using all individual words as tokens and we tried using a fixed set of pre-defined phrases (discussed in Chapter 5) as tokens.

We also tried the natural language stemming technique on the KW field so that the words: vacuole and vacuoles produce the same token.

4.1.6 Experimental Results

We used the same dataset we showed in Chapter 2 for these experiments. This dataset consists of 3218 Gram-negative bacteria from the Swiss-Prot database. We compared the accuracy of using different numbers of PSI-BLAST iterations, different numbers of top homologs, different Swiss-Prot fields and different feature extraction techniques. Table 4.1 shows the results with the most accurate choice at the top. Here is a summary of our findings about selecting homologs and extracting features:

1. Using more than one iteration of PSI_BLAST degrades the accuracy, so BLASTP can be used instead of PSI_BLAST.
2. Using the top 3 homologs is better than using the top 2 or 4 hits, although there is not a significant difference.
3. Using the SCELL, IPR and KW fields of the Swiss-Prot database altogether

gives the best prediction results, although the IPR field is the least important for predicting sub-cellular localization.

4. Using semi-colon delimited phrases for the KW field and pre-defined phrases for the SCELL field are the most accurate field extraction techniques, but not by a large margin.

It is interesting to note that for classification of other protein attributes, the results are much different. For example, a classifier that predicts the sub-classes of k^+ion channel proteins is very much dependent on the IPR field [28].

4.2 Feature Selection - Wrappers

4.2.1 Introduction

When the extraction strategies discussed in the previous section are used, PA-SUB extracts thousands of tokens from the Swiss-Prot database. However, standard machine learning algorithms do not scale well with a large number of features. Some features, which we call **common features**, like *Complete proteome* and *3D-structure* appear very often, and have little predictive value. Therefore, they are put into a list of stop-features (stop-words [29]) that are immediately filtered from the feature sets during training.

Since manually selecting and filtering features is tedious and error-prone, we investigated an automatic feature selection technique called a *feature wrapper* [18] to improve the accuracy of the classifiers. In general, after buliding and computing the accuracy of a classifier using all of the potential features, we used feature wrapping to filter some of the potential features that have the lowest *information content* [20].

4.2.2 The Information Content

The information content is computed for each feature generated by the set of training sequences. It is a measure of the amount that a feature contributes to classifications in general [20]. Informally, it tells how well that feature differentiates between different classes in the ontology. For example, if a feature is in every training instance, it is useless in discriminating between classes and its information content is zero. On the other hand, a feature that is in all training instances that have a single class label and no training instances with any other labels, is very good for discriminating the one label. Therefore, it has high information content.

Table 4.2: The information content of eight features ($F_1 ... F_8$) in eight sequences. Each class has two protein instances. Feature values are either '1' or '0' in a sequence. Information content of each feature shown in the last row is computed using Formula 4.1.

Sequence No	Class	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
1	cyt	1	1	1	0	0	0	0	0
2	cyt	1	1	1	1	0	0	0	0
3	wal	0	0	1	0	0	1	0	1
4	wal	0	0	1	0	0	0	1	1
5	ext	1	1	1	0	0	0	0	0
6	ext	1	1	1	0	0	0	0	0
7	mem	0	0	1	0	1	1	0	0
8	mem	0	1	1	1	0	0	0	0
Info. Content		.352	.352	0	.156	.147	.102	.147	.406

Given the instances, we could compute the information content I_j for each feature F_j by the Formula 4.1:

$$I_j = \sum_{c_i} \sum_{f_k} P(C = c_i, F_j = f_k) log \frac{P(C = c_i, F_j = f_k)}{P(C = c_i)P(F_j = f_k)} \tag{4.1}$$

Here c_i varies over the different ontological classes (4 in the example shown in Table 4.2, so $c_1 \ldots c_4$ represents four different classes) and f_k varies over the different values that each feature could have (2 in our case since each feature has the value of either one or zero, i.e, $f_1 = 0$ and $f_2 = 1$). Take feature F_3 as an example, since its corresponding token appears in every sequence of every class, its information content I_3 is zero, which means it is the least discriminate feature among all eight features. Feature F_8 has the highest information content, $I_8 = .406$ among these eight features since it distinguishes all the instances with a single class (wal).

4.2.3 The Wrapper Model

We used a wrapper [18] to select the subset of all the features, whose information content is above a threshold. When PA-SUB trains a classifier, it actually trains a series of classifiers with different wrapper thresholds. The first classifier is trained using all the potential features. Using 5-fold cross-validation to measure its accuracy, PA-SUB then removes 5% of the lowest information content features and builds a second classifier. Each subsequent classifier is built after removing another additional 5% of the lowest information content features. In the end, we have computed

Table 4.3: A comparison of the 5-fold cross-validation accuracies before and after using feature selection technique, the wrapper model, with different machine learning algorithms. The percentage of used features is given. It used the same dataset shown in Table 2.1.

Wrapping to select features?	NB	TAN	ANN	SVM
NO	.931	.928	.950	.939
YES	.932 at 40%	.932 at 50%	.956 at 60%	.939 at 100%

the accuracy of 20 different classifiers with 100%, 95%, 90%, $\cdots$, 5% of the original set of features. Finally, we select the classifier with the highest accuracy.

Our classifiers automatically evaluate the effect of wrapping. For example, the unwrapped Naïve Bayes classifier had an accuracy of 93.1% on the 3218 Gram-negative bacteria proteins shown in Table 2.1. The best accuracy of 93.2% was achieved by retaining 40% of the features.

The most accurate classifiers for sub-cellular localization often had 15%-25% of the features retained during our experiments. Without wrapping, classifiers often *overfit* the training data. Simply speaking, classifiers perform perfectly on the training data but poorly over the entire instances (i.e., including instances beyond the training set). However, the optimal wrapping threshold varies for different datasets and different machine learning techniques. Each of the machine learning techniques had different improvements to prediction accuracy when wrappers were used. Compared to the NB classifier with an insignificant accuracy improvement, from 93.1% to 93.2%, the TAN classifier improved 0.4%. The wrapping results are shown in Table 4.3. This table also shows that the wrapper model did not help at all when applied to the Support Vector Machine(SVM) classifier. However, even in the case where the wrapper model did not significantly increase accuracy, it is still very helpful, since it simpifies the prediction explanation process [27], as fewer features to deal with.

4.3 Summary

This Chapter discussed the way we extracted features from the Swiss-Prot database and selected the most discriminating features using the wrapping technique. It discussed our strategies of choosing the best features to achieve the best accuracy.

Chapter 5

Experiments

This chapter presents the experimental results. These results show that the classifiers we constructed are the most accurate sub-cellular localization predictors over the broadest range of organisms ever published. It also compares the accuracy of our predictors to other published prediction methods. The chapter starts by describing the way we constructed the reliable dataset from the Swiss-Prot database.

5.1 The Dataset

After selecting the ontologies shown in Table 1.2 of Chapter 1, we required oracular data for organisms in each of the categories. We constructed the dataset by extracting the proteins from Swiss-Prot release 41 [4]. Although the Swiss-Prot database contains a sub-cellular localization field(SCELL), this field does not contain a single ontological label for each sequence. There were two problems with this dataset:

- First, if the SCELL is empty for a sequence, then the sequence cannot be used for training, since we cannot assign a label.
- Second, even if the field is not empty, it often contains a long text string instead of a simple ontological label. For example, one SCELL field contains the string "secreted and associates with the extracellular matrix".

As a result, we had to construct a parser that extracted a single ontological label, when possible. For example, the parser inferred "extracellular" for the sequence with SCELL field "secreted and associates with the extracellular matrix". Here are the rules that our parser uses to label potential training sequences:

1. Check to see if the field contains one of the ontological labels as a sub-string. If it does not, the sequence is rejected as a training sequence.

2. If the field contains more than one ontological label as a sub-string, the label is *ambiguous* so it is also rejected, unless one label is an organelle and the other is "membrane".

3. If it contains an ontological label, but also contains the phrase "potential" or "by similarity" it is rejected if the number of training sequences with that label is high. However, if the number of training sequences with that label is low (<1.5% of the total number of training sequences), it is accepted.

4. If the ontological label is "cell wall" and the phrase contains the word "attached" it is rejected.

5. All proteins labelled as "fragment" are rejected.

Steps 2, 3 and 4 require some explanation.

- For step 2, it is common to describe a protein as being in a membrane of a specific organelle. In this case, the correct label is the organelle. For instance, a protein with "mitochondrial inner membrane" will be labeled as 'mitochondria'.

- Step 3 is complicated because we would like to be conservative with our training data. Therefore, we want to reject any annotations that contain words like "potential" or "by similarity" in favor of annotations that have been verified in a laboratory. However, for ontological labels with low numbers of training instances, we found that accepting "higher risk" annotations is necessary to obtain enough training data so that the classifiers have good accuracy. In addition, we followed the PSORT-B [13] lead of including any sequences that contain the phrase "cell wall" in the extracellular class for the plant and fungi kingdoms, since the Swiss-Prot data is not very accurate in these cases. An alternative would be to add a separate "cell wall" label to their ontologies.

- For step 4, we found many Swiss-Prot SCELL annotations for proteins that are not in the cell wall, contain the phrase "attached to the cell wall". Therefore, we needed a mechanism to remove this misinformation.

Following the rules above, we constructed the final training datasets summarized in Table 5.1. Full distribution of the dataset across each ontological label can be

Table 5.1: The number of training sequences are extracted by following the rules across five different categories.

Categories	Animal	Plant	Fungi	GP Bacteria	GN Bacteria
No of instances	15549	3293	2094	3174	1541

seen in Table 5.3 to Table 5.7. All of the training instances are publicly available at: http://www.cs.ualberta.ca/~bioinfo/PA/Subcellular. Each training set contains a set of sequences in FastA format that includes the correct label (extracted from Swiss-Prot), the organism tag, the organism name, some Swiss-Prot taxonomy information and the primary sequence.

5.2 k Nearest Neighbors

As described in section 2.1, a 1-nearest-neighbor classifier finds the top homolog in the Swiss-Prot database and looks the SCELL field of the homolog to extract the sub-cellular localization label. However, most proteins in the Swiss-Prot database do not have unambiguous SCELL fields (only 37% are unambiguous).

This approach can be extended to the k nearest neighbor algorithm described in section 3.2, where k specifies how many best homologs we select for inferring the label of the query protein. For example, 1NN, the simplest case, uses the sub-cellular localization from the top homologous protein directly as a prediction. 3NN (i.e., $k = 3$) selects the top three homologs and takes the majority class of the sub-cellular field as a prediction. In case all three homologs disagree on the sub-cellular

Table 5.2: A comparison of the accuracy of three nearest neighbor classifiers (1NN, 3NN and 5NN) and the Naïve Bayes (NB) classifier on the five categories of Swiss-Prot data, the LOCkey [22] and the PSORT-B [13] data.

Category	1NN	3NN	5NN	NB
Animal	.914	.923	.923	.929
Plant	.904	.915	.914	.938
Fungi	.728	.762	.754	.814
GP bacteria	.815	.848	.847	.932
GN bacteria	.870	.901	.897	.912
LOCkey	.720	.763	.768	.925
PSORT-B	.615	.652	.653	.915

field, we choose the label from the one that is most similar to the query sequence (the smallest E-Value).

A comparison of a 1NN classifier with an NB classifier was presented in Chapter 3, using a GN bacteria dataset. To evaluate the effect of using different k, we also did 3NN and 5NN experiments. The results are shown in Table 5.2. In general, 3NN is slightly better than 1NN in terms of the accuracy. This is because more homologs are selected so that the chance of getting no sub-cellular location information is reduced. Table 5.2 also shows that 5NN produced results similar to 3NN and that all the kNN results are significantly worse than the ones produced by the NB classifier.

5.3 PA-SUB Accuracy

In PA-SUB, we constructed five different NB classifiers, each for one of the five categories. Table 5.3 to Table 5.7 show the statistics for the five sub-cellular localization classifiers we built using training instrances from the Swiss-Prot database.

In general, these classifiers show excellent 5-fold cross-validation precision and recall over all ontological classes. However, some small training and test sets do produce poor results, like the 5-fold cross validation precision of vacuole on Fungi (.600). It has only 19 sequences, the smallest training/test set of all the training sets across all five categories.

Table 5.3: Confusion matrix for the PA-SUB **animal** classifier, trained on Swiss-Prot data.

Prd⇒ Obs⇓	nuc	mit	cyt	ext	gol	pex	end	lys	mem	no_pred	SUM^{obs}	recall
nuc	**2575**	2	100	11	2	0	3	0	28	125	2846	.905
mit	2	**1162**	18	4	0	3	2	0	2	5	1198	.970
cyt	47	16	**1695**	29	18	5	4	2	12	17	1845	.919
ext	2	0	23	**3655**	1	0	4	15	150	93	3943	.927
gol	1	0	5	2	**149**	0	4	1	4	1	167	.892
pex	0	2	0	0	0	**100**	0	0	1	0	103	.971
end	0	1	4	6	1	10	**435**	0	4	2	457	.952
lys	0	0	4	0	0	0	0	**161**	4	1	170	.938
mem	2	11	109	59	26	2	49	8	**4519**	35	4820	.938
SUM^{prd}	2629	1194	1958	3761	207	110	501	187	4724	279	15549	
precision	.979	.973	.866	.972	.725	.909	.868	.861	.957			**.929**

Table 5.4: Confusion matrix for the PA-SUB **plant** classifier, trained on Swiss-Prot data.

Prd⇒ Obs⇓	nuc	mit	cyt	ext	gol	chl	pex	end	vac	mem	no_pred	SUM^{obs}	recall
nuc	**162**	0	1	0	1	0	0	1	0	0	3	170	.964
mit	0	**281**	6	1	0	9	1	0	0	0	9	307	.915
cyt	1	4	**429**	2	0	7	0	0	1	0	3	447	.960
ext	0	0	1	**107**	0	1	0	0	1	0	15	127	.843
gol	0	0	0	0	**34**	0	0	0	0	0	1	35	.971
chl	1	18	23	2	1	**1818**	1	1	1	17	16	1899	.957
pex	0	1	0	0	0	0	**28**	0	0	0	0	29	.966
end	0	0	1	0	0	0	0	**57**	2	3	1	64	.891
vac	0	0	1	6	2	0	0	1	**70**	1	1	82	.854
mem	0	1	2	1	3	16	0	1	2	**102**	7	135	.756
SUM^{prd}	164	305	464	119	41	1851	30	61	79	123	56	3293	
precision	.988	.921	.925	.899	.829	.982	.933	.934	.886	.829			**.938**

Table 5.5: Confusion matrix for the PA-SUB **fungi** classifier, trained on Swiss-Prot data.

Prd⇒ Obs⇓	nuc	mit	cyt	ext	gol	pex	end	mem	vac	no_pred	SUM^{obs}	recall
nuc	**516**	4	27	4	1	1	0	2	0	66	621	.831
mit	8	**300**	39	3	0	5	1	5	0	45	406	.739
cyt	23	23	**319**	4	3	6	3	2	3	9	395	.808
ext	0	3	3	**146**	1	1	1	2	3	11	171	.854
gol	0	0	3	0	**42**	0	2	1	0	4	52	.808
pex	0	6	1	0	0	**56**	0	0	0	1	64	.875
end	2	0	2	1	6	0	**42**	7	0	4	64	.656
mem	4	1	7	4	6	1	6	**260**	2	11	302	.861
vac	0	0	2	2	2	0	1	0	**12**	0	19	.632
SUM^{prd}	553	337	403	164	61	70	56	279	20	151	2094	
precision	.933	.890	.792	.890	.689	.800	.750	.932	.600			**.814**

Table 5.6: Confusion matrix for the PA-SUB **Gram-positive bacteria** classifier, trained on Swiss-Prot data.

Prd⇒ Obs⇓	cyt	cew	ext	mem	no_pred	SUM^{obs}	recall
cyt	**881**	0	26	13	10	930	.947
cew	1	**16**	1	0	1	19	.842
ext	4	2	**217**	8	21	252	.861
mem	8	1	17	**291**	23	340	.856
SUM^{prd}	910	20	329	312	55	1541	
precision	.985	.842	.831	.933			**.912**

Table 5.7: Confusion matrix for the PA-SUB **Gram-negative bacteria** classifier, trained on Swiss-Prot data.

Prd⇒ Obs⇓	cyt	ext	per	inn	cew	out	no_pred	SUM^{obs}	recall
cyt	**1775**	21	26	12	0	4	23	1861	.954
ext	3	**216**	7	0	2	1	24	253	.854
per	5	14	**333**	5	0	6	22	385	.865
inn	5	1	4	**411**	0	0	11	432	.951
cew	0	1	0	0	**43**	1	1	46	.935
out	0	4	2	2	0	**181**	8	197	.919
SUM^{prd}	1788	257	372	430	45	193	89	3174	
precision	.993	.840	.895	.956	.956	.938			**.932**

5.4 *1-Organism* Classification

PA-SUB was developed for the purpose of predicting the sub-cellular localizations of all sequences in a newly sequenced organism. In such a case, however, we cannot evaluate the accuracy of our predictions since we do not know the correct labels beforehand. Therefore, to simulate this situation, we designed an experiment called a "1-organism classifier". Basically, we built a single NB classifer from all training data except the sequences from one specific organism. Next, we evaluated this classifier by applying it to the sequences of the organism that was not in the training process. For fairness, all Swiss-Prot entries for that specific organism were removed from the Swiss-Prot database.

We selected five different organisms across five differnt categories for 1-Organism experiments to show the prediction ability of our classifiers. They are: Bos taurus (animal), Zea mays (plant), neurospora crassa (fungi), haemophilus influenzae (Gram-positive bacteria) and Streptomyces coelicolor (Gram-negative bacteria). The basic rules for choosing the 1-organism were:

- The organism should not be the most common in its category. For example, we cannot evaluate the Gram-negative bacteria classifier effectively if we selected *E.Coli* as our 1-organism because *E.Coli* sequences are the most common GN bacteria in Swiss-Prot and were used to infer the function of many other sequences in Swiss-Prot.
- The organism must have a good coverage across the ontological labels in its category. Ideally, the organism should contain sequences of each ontological label so that we could evaluate the classifer across all the labels based on precision and recall numbers. However, in some case, it is hard to find such an organism so we chose the one with as many classes as possible. For instance, both the 1-organisms we selected for GP and GN lack the 'cell wall' class.

Table 5.8 to Table 5.12 show excellent results for all five 1-Organism classifers.

Table 5.8: Confusion matrix for the 1-organism classifier, trained on animal dataset from Swiss-Prot. The 1-organism is **Bos taurus (Bovine)**.

Prd⇒ Obs⇓	nuc	mit	cyt	ext	gol	pex	end	lys	mem	SUM^{obs}	recall
nuc	**42**	0	3	2	0	0	0	0	0	47	.894
mit	0	**138**	1	6	0	0	0	0	0	145	.952
cyt	0	0	**79**	2	2	1	0	0	0	84	.940
ext	0	0	0	**190**	0	0	0	1	6	197	.964
gol	0	0	0	0	**6**	0	0	0	1	7	.857
pex	0	0	0	0	0	**4**	0	0	0	4	1.00
end	0	0	0	0	0	0	**14**	0	0	14	1.00
lys	0	0	0	0	0	0	0	**12**	0	12	1.00
mem	0	4	7	2	1	0	3	1	**200**	218	.917
SUM^{prd}	42	142	90	202	9	5	17	14	207	728	
precision	1.00	.972	.878	.941	.667	.800	.824	.857	.966		**.941**

Table 5.9: Confusion matrix for the 1-organism classifier, trained on plant dataset from Swiss-Prot. The 1-organism is **Zea mays (Maize)**.

Prd⇒ Obs⇓	nuc	mit	cyt	ext	gol	chl	pex	end	vac	mem	SUM^{obs}	recall
nuc	**16**	0	0	0	0	0	0	0	0	0	16	1.00
mit	0	**18**	0	0	0	0	1	0	0	0	19	.947
cyt	0	0	**33**	3	0	0	0	0	0	0	36	.917
ext	0	0	0	**6**	0	0	0	0	0	0	6	1.00
gol	0	0	0	0	**2**	0	0	0	0	0	2	1.00
chl	0	2	1	0	0	**63**	0	0	1	2	69	.913
pex	0	0	0	0	0	0	**1**	0	0	0	1	1.00
end	0	0	0	0	0	0	0	**6**	0	0	6	1.00
vac	0	0	0	0	0	0	0	0	**2**	0	2	1.00
mem	0	0	0	4	0	2	0	0	0	**3**	9	.333
SUM^{prd}	16	20	34	13	2	65	2	6	3	5	166	
precision	1.00	.900	.971	.462	1.00	.969	.500	1.00	.667	.600		**.904**

Table 5.10: Confusion matrix for the 1-organism classifier, trained on fungi dataset from Swiss-Prot. The 1-organism is **Neurospora crassa**.

Prd⇒ Obs⇓	nuc	mit	cyt	ext	gol	pex	end	mem	vac	SUM^{obs}	recall
nuc	**10**	0	0	0	0	0	0	1	0	11	.909
mit	0	**40**	2	3	0	0	0	0	0	45	.889
cyt	0	1	**14**	0	0	0	0	0	0	15	.933
ext	0	0	0	**2**	0	0	0	0	0	2	1.00
gol	0	0	0	0	**0**	0	0	0	0	0	-
pex	0	0	0	0	0	**0**	0	0	0	0	-
end	0	0	0	0	0	0	**1**	0	0	1	1.00
mem	0	0	1	0	0	0	0	**11**	0	12	.917
vac	0	0	0	0	0	0	0	0	**1**	1	1.00
SUM^{prd}	10	41	17	5	0	0	1	12	1	87	
precision	1.00	.976	.824	.400	-	-	1.00	.917	1.00		**.908**

Table 5.11: Confusion matrix for the 1-organism classifier, trained on Gram-positive bacteria dataset Swiss-Prot. The 1-organism is **Haemophilus influenzae**.

Prd⇒ Obs⇓	cyt	wal	ext	mem	SUM^{obs}	recall
cyt	**37**	0	0	0	37	1.00
cew	0	**0**	0	0	0	-
ext	0	0	**6**	0	6	1.00
mem	0	0	1	**8**	9	.889
SUM^{prd}	37	0	7	8	52	
precision	1.00	-	.857	1.00		**.981**

Table 5.12: Confusion matrix for the 1-organism classifier, trained on Gram-negative bacteria from Swiss-Prot. The 1-organism is **Streptomyces coelicolor**.

Prd⇒ Obs⇓	cyt	ext	per	inn	wal	out	SUM^{obs}	recall
cyt	**73**	0	0	0	0	0	73	1.00
ext	0	**13**	0	0	2	0	15	.867
per	0	0	**7**	0	0	0	7	1.00
inm	0	0	1	**4**	0	0	5	.800
cew	0	0	0	0	**0**	0	0	-
oum	0	6	0	0	0	**9**	15	.600
SUM^{prd}	73	19	8	4	2	9	115	
precision	1.00	.684	.875	1.00	0	1.00		**.922**

5.5 Comparisons with Other Methods

We performed some specific additional experiments to compare our work with others. Specifically, we compared our classification technique to the Swiss-Prot lexical technique used in LOCkey [22] and built a custom classifier for GN bacteria using the reliable PSORT-B data [13] and compared it to the PSORT-B predictor.

5.5.1 Comparion with LOCkey

To compare with LOCkey, we constructed two custom sub-cellular localization classifiers using their single ontology and their training data. The LOCkey paper [22] contains a confusion matrix for a Swiss-Prot sub-set that contains 1161 training sequences for which they achieved the best accuracy (Note this is only 37% coverage of a larger 3146 sequence set). Table 5.13 shows the five-fold cross-validation precision, recall and accuracy computed from their confusion matrix and from a PA-SUB classifier we built using their training data and their ontology. Our precision and accuracy results are consistently better than the ones they reported, except for the recall on the golgi class. Our overall accuracy is almost 10% better at .922 versus .815. Even though their approach is similar to PA-SUB, there are two reasons for these accuracy differences.

1. We are using a different classifier technology - Naïve Bayes versus ad-hoc.
2. We are using different Swiss-Prot database fields (PA-SUB uses the IPR field).

Their paper does not include a confusion matrix or accuracy statistics, for 100% coverage of the 3146 sequences set, other than to indicate that the accuracy is less than the .815 accuracy of their 37% coverage classifier. On this larger set (100% coverage) we achieved an accuracy of .890 (Table 5.14).

Table 5.13: A comparison of the statistics of a PA-SUB custom classifier built using the LOCkey 1161 sequence training data with the statistics produced by their classifier on their training data.

location	count	LOCkey		PA-SUB	
		precision	recall	precision	recall
nuc	350	.850	.971	.965	.979
end	13	.200	.154	1.000	.500
gol	21	.895	.810	.944	.773
lys	6	.000	.000	.833	.714
mit	190	.763	.795	.964	.979
per	9	.000	.000	1.000	.375
chl	92	.718	.609	.966	.894
vac	5	.000	.000	1.000	.250
cyt	138	.656	.428	.804	.846
ext	337	.879	.953	.893	.973
Overall	1161	acc.→	.815	acc.→	.922

Table 5.14: Confusion matrix for a custom NB classifier, trained on LOCkey 3146 protein sequences (100% coverage of test data).

Prd⇒ Obs⇓	mit	ext	nuc	chl	cyt	end	lys	gol	pex	vac	SUM^{obs}	recall
mit	**17**	0	1	0	0	1	1	3	4	1	28	.607
ext	0	**52**	0	0	0	1	0	2	0	0	55	.945
nuc	1	0	**842**	1	1	5	2	47	21	2	922	.913
chl	0	0	0	**182**	0	2	0	10	2	1	197	.924
cyt	0	0	1	0	**44**	0	1	3	0	0	49	.898
end	0	1	3	4	0	**436**	0	15	8	0	467	.934
lys	1	0	0	0	1	0	**42**	2	5	1	52	.808
gol	1	4	24	5	0	8	4	**463**	28	7	544	.851
pex	2	0	14	2	4	6	8	48	**629**	11	724	.869
vac	1	0	3	0	0	0	3	6	2	**93**	108	.861
SUM^{prd}	23	57	888	194	50	459	61	599	699	116	3146	
precision	.739	.912	.948	.938	.880	.950	.689	.773	.900	.802		**.890**

Table 5.15: A comparison of the statistics of a PA-SUB custom classifier built using the PSORT-B training data with the statistics produced by the PSORT-B predictor, built from the same training data. The ontological labels are: (cyt)oplasmic, (inn)er membrane, (per)iplasmic, (out)er membrane and (ext)racellular.

location	count	PSORT-B		PA-SUB	
		precision	recall	precision	recall
cyt	252	.976	.694	.947	.857
inn	308	.967	.787	.972	.912
per	264	.919	.576	.916	.867
out	378	.988	.903	.984	.952
ext	241	.944	.700	.755	.971
Overall	1443	acc.→	.748	acc.→	.915

5.5.2 Comparison with PSORT-B

Table 5.15 shows the five-fold cross-validation precision, recall and accuracy presented in PSORT-B [13] paper and the same statistics computed from a PA-SUB custom classifier built using the PSORT-B training data and ontology. Note that our Swiss-Prot GN bacteria ontology has one extra label, (cell)wal(l), which they included in the ext(racellular) class. To compare our technique more directly with theirs, we did not include a (cell)wal(l) label in the classifier we built from their data. Our approach is very different than theirs, since we use a simple Naïve Bayes classifier and features extracted from Swiss-Prot homologs, while they use a set of six analytical models. Nevertheless, our approach produces results that are quite comparable in precision, but significantly better in recall and overall accuracy (from .748 to .915).

Note that 139/1443 training sequences in the PSORT-B training data have more than one location. To accommodate dual-labels in our Naïve Bayes classifier, we transformed each training instance that had two labels into two training instances, one with each label. Since we are comparing with the PSORT-B classifier [13], we followed their lead during predictor evaluation and counted a prediction as correct if it predicted either of the two labels.

5.6 Summary

This chapter described the way we developed the training datasets from the Swiss-Prot database. It presented the excellent results of PA-SUB classifiers, built using

the five different datasets, one for each category. It also compared our technique with two competitive predictors, "LOCkey" and "PSORT-B". We conclude in the next chapter and give some future research directions.

Chapter 6

Discussion and Future Work

Section 6.1 introduces the coverage problem PA-SUB currently has. Section 6.2 discusses some of the future directions, including one that might solve the coverage problem. Section 6.3 summarizes the contributions of this research.

6.1 PA-SUB Coverage

The PA-SUB accuracy results reported in this dissertation are shown in Table 5.3 to Table 5.7. The no_prediction column of these tables can be used to compute coverage across the five different categories, ranging from .928 to .982 as shown in Table 6.1. There are two reasons for not making a prediction: Firstly, there will be some sequences for which there are no homologs, so no tokens can be extracted and used by the classifier. Secondly, in some cases, even though homologs are found, there will be no relevant tokens in the KW, IPR and SCELL fields that were used by PA-SUB to construct the token set of the classifier. In either case, we call such sequences *excluded sequences*. PA-SUB makes no sub-cellular localization prediction for excluded sequences. Excluded sequences are the only ones that reduce the coverage of PA-SUB classifiers.

Table 6.1: The current PA Coverage across five different categories. The number of instances and no_prediction are from Table 5.3 to Table 5.7 of Chapter 5.

Categories	Animal	Plant	Fungi	GP Bacteria	GN Bacteria
No of instances	15549	3293	2094	1541	3174
No of no_prediction	279	56	151	55	89
Coverage	.982	.983	.928	.964	.972

However, the coverage numbers shown in Table 6.1 may be misleading since the

Table 6.2: Coverage of the PA-SUB classifier on some fully sequenced organisms. The count is the total number of genetic sequences for the organism. An excluded sequence (excluded) is one for which PA-SUB was unable to find at least one homolog whose E-value was less than .001 that contained at least one relevant feature. Coverage is the percentage of all non-excluded sequences from an organism.

organism	category	count	exclude	coverage
Mus musculus	animal	27754	7099	.745
Guillardia thetá	animal	451	159	.647
Arabidopsis thaliana	plant	26032	10043	.600
Schizosaccharo myces pombe	fungi	5007	1023	.787
Streptomyces coelicolor	GP bacteria	8038	3305	.589
Bacillus subtilis	GP bacteria	4098	1346	.672
Pseudomonas aeruginosa	GN bacteria	5557	1355	.756
E.coli K-12	GN bacteria	4357	922	.788

sequences used are all from Swiss-Prot. To gain an appreciation for the PA-SUB coverage on various organisms, we used the PA-SUB classifiers to classify all of the sequences in several organisms. This includes many sequences that are not in Swiss-Prot. As we did in the *1-O classifier* experiments, for fairness, we removed all of the sequences for that organism from Swiss-Prot before running PA-SUB on it. Therefore, no exact sequence matches would be found. The results are shown in Table 6.2. Of course, for these tests, we cannot report accuracy, since we included all sequences from the organisms, and we do not know the "correct" sub-cellular localization for many of them. The organisms use the animal, plant, fungi, GN bacteria and GP bacteria classifiers respectively. Each organism was selected since its complete proteome is publicly available.

6.2 Future Directions

6.2.1 Pattern Discovery

We are currently developing pattern discovery algorithms to allow us to classify excluded sequences. The idea is to extract *local features* directly from the primary sequences for each excluded sequence. One example of local features are *motifs* (*protein patterns*), which are regions of protein sequences that are of functional significance. These motifs may range in size from only a few amino acids in length to large structural domains which dominate the entire length of a sequence. Using motifs as features would allow us to give a prediction of sub-cellular localization on

a complete proteome basis, without the need for annotated homologs.

6.2.2 Gene Ontology Classification Scheme

Instead of using the current ontologies for each of our five categories, we are expanding our ontologies to cover a number of biologically as well as pharmaceutically interesting categories in the Gene Ontology (GO) classification system [7]. GO contains 1305 cellular component terms covering sub-cellular structures, locations, and macromolecular complexes. Examples include nucleus, telomere, and origin recognition complex. As a result, it provides a more specific description of the sub-cellular localization than our current five ontologies do. There are two challenges: first, we need to develop a good dataset. It is impossible to directly obtain a large set of sequences annotated according to the GO classification scheme. Second, we need to investigate the best classification technique for this problem in terms of accuracy and efficiency.

6.2.3 Parallelism

To build a classifier with thousands of training sequences, the system currently takes more than one day. Half of the time is dedicated to PSI-BLASTing against Swiss-Prot. Therefore, improving the speed of PSI-BLAST will be important in building new classifiers and expermenting with new parameters. One approach is to integrate a parallel technique into the system since each PSI-BLAST is independent so that they could be easily parallelized.

6.3 Contributions

We have succeeded in developing PA-SUB, a web-based tool for high-throughput prediction of protein sub-cellular localization. We described the architecture of the system, introduced several key strategies like 'feature extraction' and presented the results of the experiments we have designed. Our results show that the technique has made the most accurate sub-cellular localization predictions over the broadest range of organisms, compared with any other published results. In addition, since the technique has proven successful for sub-cellular localization prediction, it can probably be used for protein property prediction in general.

Bibliography

[1] SF Altschul, W Gish, W Miller, EW Myers, and DJ Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–10, 1990.

[2] SF Altschul, TL Madden, AA Schaffer, J Zhang, Z Zhang, W Miller, and DJ Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–402, 1997.

[3] A.Reinhardt and T.Hubbard. Using neural networks for prediction of the subcellular location of proteins. *Nucleic Acids Research*, 26(9):2230–2236, 1998.

[4] B. Boeckmann, A. Bairoch, R. Apweiler, M.-C. Blatter, A. Estreicher, E. Gasteiger, M.J. Martin, K. Michoud, C. O'Donovan, I. Phan, S. Pilbout, and M. Schneider. The swiss-prot protein knowledgebase and its supplement trembl in 2003. *Nucleic Acids Research*, 31:365–370, 2003.

[5] Juan Cedano, Patrick Aloy, Josep A. Pérez-Pons, and Enrique Querol. Relation between amino acid composition and cellular location of proteins. *Journal of Molecular Biology*, 266:594–600, 1997.

[6] C.K. Chow and C.N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transaction On Information Theory*, 14:462–467, 1968.

[7] The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genet.*

[8] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, 1973.

[9] Frank Eisenhaber and Peer Bork. Wanted: subcellular localization of proteins based on sequence. *Trends in Cell Biology*, 8(4):169–170, 1998.

[10] Olof Emanuelsson, Søren Brunak, and Gunnar von Heijne. Chlorop, a neural network-based method for predicting chloroplast transit peptides and their cleavage sites. *Protein Science*, 8:978–984, 1999.

[11] Olof Emanuelsson, Henrik Nielsen, Søren Brunak, and Gunnar von Heijne. Predicting subcellular localization of proteins based on their n-terminal amino acid sequence. *J.Mol.Biol.*, 300(5):1005–1016, 2000.

[12] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

[13] Jennifer L. Gardy, Cory Spencer, Ke Wang, Martin Ester, Gábor E. Tusnády, István Simon, Sujun Hua, Katalin deFays, Christophe Lambert, Kenta Nakai, and Fiona S.L. Brinkman. PSORT-B: Improving protein subcellular localization prediction for gram-negative bacteria. *Nucleic Acids Research.*, 31(13):1–5, 2003.

[14] Paul Horton and Kenta Nakai. A probabilistic classification system for predicting the cellular localization sites of proteins. In *Intelligent Systems for Molecular Biology 4*, pages 109–115, 1996.

[15] Paul Horton and Kenta Nakai. Better prediction of protein cellular localization sites with the k nearest neighbors classifier. In *Intelligent Systems for Molecular Biology 5*, pages 147–152, 1997.

[16] Sujun Hua and Zhirong Sun. Support vector machine approach for protein subcellular localization. *Bioinformatics*, 17(9):721–728, 2001.

[17] K.Nakai and M.Kanehisa. A knowledge base for predicting protein localization sites in eukaryotic cells. *Genomics*, 14:897–911, 1992.

[18] R. Kohavi and G.H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97:273–324, 1997.

[19] DJ. McGeoch. On the predictive recognition of signal peptide sequences. *Virus Res*, 3(3):271–86, 1985.

[20] Tom Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[21] Klaus-Robert Muller, Sebastian Mika, Gunnar Rätsch, Koji Tsuda, and Bernhard Schölkopf. An introduction to kernal-based learning algorithms. *IEEE TRANSACTIONS ON NEURAL NETWORKS*, 12(2), 2001.

[22] Rajesh Nair and Burkhard Rost. Inferring sub-cellular localization through automated lexical analysis. In *Proceedings of the tenth International Conference on Intelligent Systems for Molecular Biology*, pages 78–86. Oxford University Press, August 2002.

[23] H. Nakashima and K. Nishikawa. Discrimination of intracellular and extracellular proteins using amino acid composition and residue-pair frequencies. *Journal of Molecular Biology*, 238(1):54–61, 1994.

[24] Henrik Nielsen, Søren Brunak, and Gunnar von Heijne. Review: Machine learning approaches for the prediction of signal peptides and other protein sorting signals. *Protein Engineering*, 12(1):3–9, 1999.

[25] Henrik Nielsen, Jocob Engelbrecht, Søren Brunak, and Gunnar von Heijne. Identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites. *Protein Engineering*, 10(1):1–6, 1997.

[26] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1995.

[27] Duane Szafron, Russ Greiner, Paul Lu, David Wishart, Cam MacDonell, John Anvik, Brett Poulin, Zhiyong Lu, and Roman Eisner. Explaning naïve bayes classifications. Technical report, Dept. of Comp. Science, U. of Alberta, 2003.

[28] Duane Szafron, Paul Lu, Russ Greiner, David Wishart, Zhiyong Lu, Brett Poulin, Roman Eisner, John Anvik, Cam Macdonell, and B.Habibi-Nazhad. Proteome analyst - transparent high-throughput protein annotation: Function, localization and custom predictors. In *ICML 2003 Workshop: Machine Learning in Bioinformatics*, http://www.cs.ualberta.ca/~bioinfo/PA, August 2003.

[29] C.J. van Rijsbergen. *Information Retrieval.* Butterworths, London(UK), 1979.

[30] von Heijne G. A new method for predicting signal sequence cleavege site. *Nucleic Acids Research*, 14(11):4683–90, 1986.